IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 2, March 2013
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org

The Implementation of Univariate and Bivariate B-Spline Interpolation Method in Continuous

Dian Pratiwi[1]

[1] **Department of Information Engineering, Trisakti University**
Jakarta, 15000, Indonesia

Abstract
There are various ways of estimating an unknown value between intervals, one of which is to apply the interpolation method. In this study, the interpolation method is implemented to find the value of new points in making the curve and B-Spline surface from several the control points that are known. In making the curve, interpolation is used in conjunction with the Bernstein polynomial basis in finding the value of basis with one parameter (u), which then produce a polynomial function y = f(u) of 3 degree. This function is then used to search for new dots around 4 pieces of control points is given. While on the surface creation, interpolation is used in conjunction with Cox de Boor algorithm to find the value of basis with two parameters (u, v) and the knot vectors between 0 and 1, which was also subsequently produced a polynomial function y = f(u, v) of 3 degree. This function is then used to get the knot points between segments are formed by 16 pieces of control points is given. Overall the result of this study, both the curve and B-Spline surface, the interpolation method worked quite well in generating the new points, due to the curvature of curve and B-Spline surface are formed very smooth.
Keywords: *Bernstein Polynomial, Cox de Boor, B-Spline, Knot Vector, Interpolation.*

1. Introduction

The interpolation is a technique commonly used to find a value that is at a certain interval [6]. In its quest, can use a variety of methods that will produce a polynomial function y = f(x). The function is then used to generate the values that we want to know [5].

In this study, the interpolation method is used to form a curve (univariate) and B-Spline surface (bivariate) of known n- control points. In the curve formation, the starting point used by 4 control points (P0, P1, P2, P3). As for the surface, which is used as a starting point 4x4 control points (P0 – P15). These points are then interpolated with B-Spline Basis to produce a new control point (the knot) that will form the curve and B-Spline surface continuously. B-Spline Basis is used to follow Cox de Boor algorithm which evaluation starts from 4 control points and then generalizations every one of control point.

Detail point of the curve and B-Spline surface is dependent of the number of interpolation is done. The more of detail (level of detail) or the number of curve and surface knots is desired, then the interpolation is needed more and more also.

2. B-Spline Interpolation

The curve is a picture generated by interpolation from granting control points to a certain amount which is generally in the form of a curvy line. Curves and surfaces are formed depends on several factors, namely the method of interpolation, the degree or amount of initial control points and basis functions are used.

Interpolation is a technique in finding the value estimate contained in a certain interval. Interpolation on the curve will result in new control points, which in this case is called the knots.

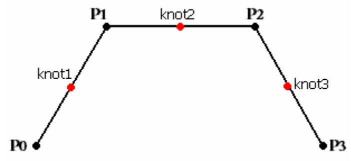


Fig. 1 The Formation of 3 Knots from 4 Control Points are known

In figure 1, the control points are used to produce a line called Convex Hull. The line is an area of the curve or surface to be formed when the interpolation is done, so the shape of the curve or surface will depend on the initial control points are given.

2.1 B-Spline Curve

In 1959, Paul de Faget de Casteljau introduced de Casteljau algorithm as a way to calculate and recursive the polynomial function of Bezier or Bernstein curve. And B-Spline curve is an extension of the Bezier curve.

The de Casteljau algorithm [12] :

$$P_{i,j} = (1-u)P_{i-1,j} + uP_{i-1,j+1} \tag{1}$$

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

where :
$P_{i,j}$ = coefficient of Bernstein polynomial
u = knots or control points
The algorithm is then called de Boor algorithm

B-Spline curve is a curve generated from the four of control points and form a short curve. This curve looks like floating and not touch the starting point (P0) and the end control point (P3).

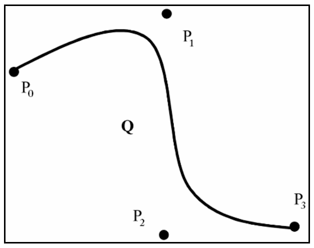


Fig. 2 B-Spline Curve with 4 Control Points

B-Spline curve is more flexible than the previous curve (Bezier). This is because the B-Spline curves are generalizing many control points at each one control point and not just limited to 4 control points only [8].

The formation of B-Spline curves can be described as follows :

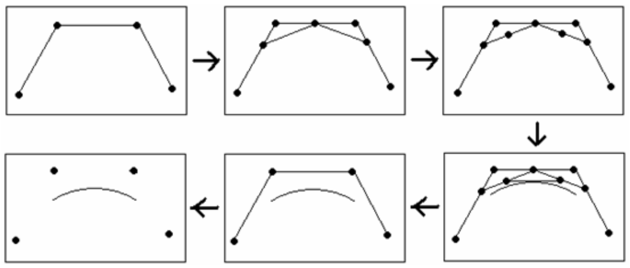


Fig. 3 The process of B-Spline Curves Formation from 4 Control Points

And generalization control points of B-Spline curve can be seen in the following table :

Table 1: Control Points Generalization

Segment	***Control Point***
3	P0, P1, P2, P3
4	P1, P2, P3, P4
5	P2, P3, P4, P5
6	P3, P4, P5, P6
…etc	….etc

From table 1 it can be seen that the B-Spline curve consists of segments, and points in between these segments are called knots.

There are several important elements in a formation of the curve and B-Spline surface, namely [1] :

- Degree

  The degree regulate how close the curve through the control points of B-Spline curve. The smaller the value of the degree, the closer the curve passes through the constituent of control points, and vice versa.

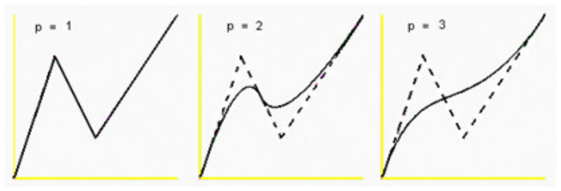


Fig. 4 B-Spline Curve with a different Degree (p)

  In figure 4 can be seen, with increasing the value of degree, the shape of the curve will be smoother.

- Basis or Blending Function

  Blending function is a function that determines how large the curvature of the B-Spline curve is influenced by the value of degree, knot vectors and control points.

- Knot Vector [11]

  Knot vector is an interval value that is used when the curve interpolation occurs. These knot vectors (w) in the cubic B-Spline curve defined by :

$$w = n + 4 \qquad (2)$$

and generalizations [11] :

$$w = n + m + 1 \qquad (3)$$

where :
n = amount of control points
m = degree
If the B-Spline curve through the start and the end control point, the knot vectors :

$$u_0 = u_1 = ... = u_m = 0 \qquad (4)$$

$$u_{m+1}, ... , u_{w-m-1}, u_{w-m} = u_{w-m+1} = ... = u_w = 1 \qquad (5)$$

more precisely :
The first *m+1* is 0, and the last value of *m+2* is 1. As for the *n-m* knots which is also called internal knots can be found by [2][9] :

- Using the calculation :

$$u_0 = u_1 = ... = u_m = 0 \qquad (6)$$

$$u_{i+m} = j / (n-m+1) \ ; \text{for } j = 1, 2, ... n-m \qquad (7)$$

$$u_{w-m} = u_{w-m+1} = ... = u_w = 1 \qquad (8)$$

so, the knot vectors are formed [9] :

$$\{0, 0, 0, 0, u_4, u_5\ 1, 1, 1, 1\} \quad (9)$$

- Find the average value of the existing control points :

$$u_0 = u_1 = ... = u_m = 0 \quad (10)$$

$$u_{i+m} = 1/m\left(\sum_{i=j}^{j+m-1} t_i\right) \quad \text{for } j = 1, 2, \dots n-m \quad (11)$$

$$u_{w-m} = u_{w-m+1} = ... = u_w = 1 \quad (12)$$

with the first knot internal is an average value of *m* parameters, namely $u_1, u_2, \dots u_m$. The second knot internal is the average value of the next *m* parameter, namely $u_2, u_3, .. u_{m+1}$.

So, if there is a *m* degree with m+1 control points (P), it can be formulated for the Bernstein polynomial function [4] :

$$W^m(u) = \sum_{i=0}^{n} P_i B_{i,m}(u) \quad (13)$$

where :

W = a new control point
P_i = *i*- th control point (i = 0.....n)
$B_{i,m}$ = *i*- th Bernstein polynomial basis, *m* degree
u = the knot
m = degree
n = amount of control points

The Basis of Bernstein polynomial in this case can be found by the formula [10][13] :

$$B_{i,m}(u) = \binom{m}{i} u^i (1-u)^{m-i} \quad (14)$$

$$B_{i,m}(u) = \frac{m!}{i!(m-i)!} u^i (1-u)^{m-i} \quad (15)$$

2.2 B-Spline Surface

A B-Spline surface is defined as the tensor product of two B-Spline curves along x and y directions [9]. As with the curves, B-Spline surface also has a degree, basis functions, knot vector as an important element in the formation. But there is a difference in the basis functions [8].

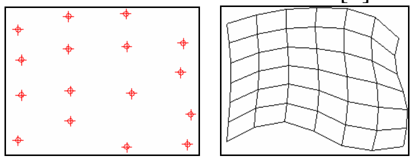


Fig. 5 Control Points of B-Spline Surface Interpolation (left) and B-Spline Surface (right)

Basis functions on the B-Spline surface is a development of the previous algorithm (de Boor), ie Cox de Boor algorithm [3][10].

$$B_{i,1}(t) = 1.0 \text{ if } t_i \le t \le t_{i+1}, \text{ others } 0.0 \quad (16)$$

$$B_{i,2}(t) = \frac{t-t_i}{t_{i+1}-t_i} B_{i,1}(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} B_{i+1,1}(t) \quad (17)$$

$$B_{i,n}(t) = \frac{t-t_i}{t_{i+n-1}-t_i} B_{i,n-1}(t) + \frac{t_{i+n}-t}{t_{i+n}-t_{i+1}} B_{i+1,n-1}(t) \quad (18)$$

where :

t = the knot
$B_{i,n}$ = *i*- th Basis control point, *n* degree

These basis function are then used to search for Bernstein polynomials by the formula [4][9] :

$$T(u,v) = \sum_{i=0}^{n} \left\{ \sum_{j=0}^{m} P_{ij} B_i^d(u) \right\} B_j^d(v) \quad (19)$$

where :

u,v = the knots
n, m = control points
$P_{i,j}$ = *i,j*- th control points (i=0...n ; j=0....m)
B = basis
d = degree
T = a new knot point

From the formula it can be concluded that for the B-Spline surface will form a parametric equation with two dependent variable (u and v), which in this case is called bivariate. As for the B-Spline curve, its parametric equations there is only one dependent variable (u) or univariate.

3. Continuous Univariate and Bivariate

Variable is a symbol or a concept that is assumed as a set of values [7]. Two or more variables can be functionally related.

In the functional relationship, there are two types of variables :

- Independent Variable

  Independent variable is a stimulus variable or variables that affect the value of other variables.
- Dependent Variable

  Dependent variable is a variable to react /respond if the independent variable associated with.

Functional relationship or function has a number of possibilities such as univariate, bivariate, trivariate, multivariate, linier and non linier.

- Univariate Function

  Is generally expressed in the form :

$$y = f(x) \quad \text{or} \quad y = g(x) \quad (20)$$

for example : y = 2x + 1

where, x is the independent variable and y is the dependent variable.
For a linier univariate function, the coordinate system in the form will be a straight graph. Example : $y = 2x + 1$. While the function of non-linier univariate, function graph that form does not show a straight line. For example : $y = 2x^2 + 1$

- Bivariate, Trivariate, Multivariate Function

  Bivariate function has two independent variables. Trivariate function have three independent variables, and multivariate function has many variables.
  Example :
  $z = 2x + 3y – 6$ (bivariate)
  $z = 5w – 3x – 2y + 1$ (trivariate)
  Just as univariate, in bivariate, trivariate and multivariate function also linier or non linier.

4. Analysis and Design

This section outlines the analysis and design of the implementation of univariate and bivariate B-Spline interpolation method in continuous in the form of B-spline basis algorithm, Cox de Boor, flowcharts, and an explanation of calculations.

4.1 B-Spline Curve

In this study, the interpolation of B-Spline curve were constructed has 3 degree or cubic B-Spline, where its formation stages as follows :

- Determining the xyz coordinates in 4 control points (P0, P1, P2, P3)

- Determining the value of B-Spline basis with Bernstein polynomial basis algorithm (equation 15) as follows :

```
degree = 3
knot = 0.4
control_point = degree + 1
While i = 0 < control_point, do {
        While m = 0 < control_point, Do {
        B[i][m] = (m! / (i! * (m-i)! )* (knot^i) *
        ((1-knot)^(m-i))
        m = m + 1
        }
i = i + 1
}
```

  The basis obtained will be 4x4 matrix
- Processing the basis value that have known to control points using the Bernstein polynomial algorithm (equation 13) as follows :

```
degree = 3
xyz_axis = 3
While i = 0 < (degree + 1), Do {
        While j = 0 <  xyz_axis, Do {
                While k = 0 < total_knot, Do {
        W[i][j] = W[i][j] + (B[i][k] * P[k][j])
        k= k + 1
        }
        j = j + 1
        }
i = i + 1
}
```

  From this step will get new points (knots) which is the stored in a matrix of size 4x3.
- The matrix that containing the new points will then be called by the OpenGL function that will automatically establish a curved line in the area around the control points that have been included.

Flowchart of the formation stages of B-Spline curve interpolation is described as follows :

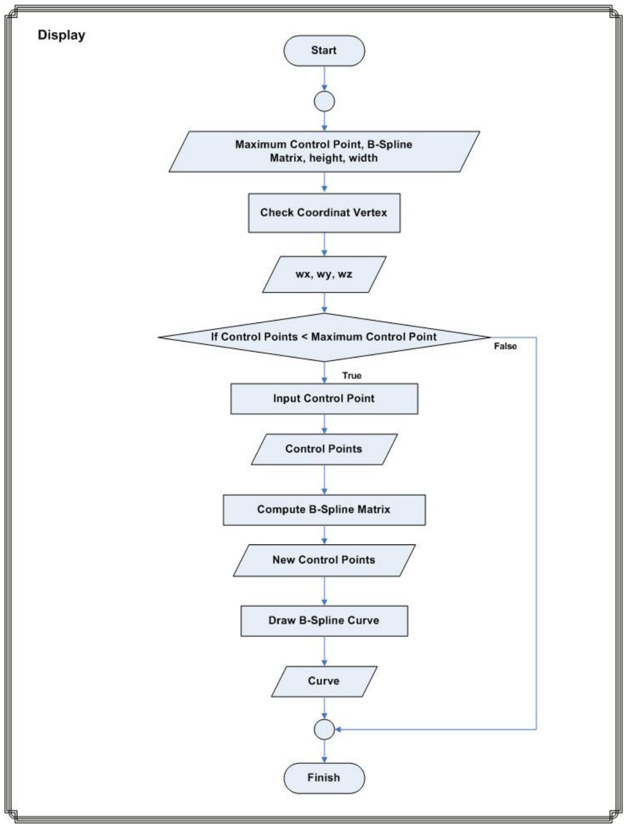


Fig. 6 Flowchart of B-Spline Curve Interpolation

4.2 B-Spline Surface

Unlike the B-Spline curve, on the B-Spline surface, the knots used as a point that appears on the segment between control points.

The knot in this study consists of u and v knot value between 0.0 and 1.0, where the intervals are stored in knots vector. If the knot value obtained exceeds the value of knot vector, the knots will appreciating 0.

The algorithm used in the manufacture of B-Spline basis on the B-Spline surface is the Cox de Boor algorithm. The stages of formation is as follows :

- Determine xyz coordinates at 16 control points (P0 to P15).
- Determine the value of knot vector (equation 2 and 3).
- Determine u and v knot of knot vectors that have been obtained.
- Determine the initial LOD (Level of Detail) or number of knot.
- Finding the value of B-Spline basis using Cox de Boor algorithm (equation 16, 17, 18). The algorithm is the following :

```
k =degree + 1
While a = 0 < degree, Do {
        If k == 1, Then {
        If knot_vector[a] <=knot and knot <
        knot_vector[a+1], Then {
                        basis_value = 1.0
                }
                        basis_value = 0.0
        }
Den1 = knot_vector[a+k-1] – knot_vector[a]
Den2 = knot_vector[a+k] – knot_vector[a+1]
Den 1 = 0
Den2 = 0
If Den1 > 0, Then {
 Eq1 = ((knot – knot_vector[a]) / Den1) *
CoxDeBoor[knot, a, k–1, knot_vector]
}
If Den2 > 0, Then {
Eq2 = ((knot_vector[a+k] – knot) / Den2) *
CoxDeBoor[knot, a+1, k-1, knot_vector]
}
        basis_value = Eq1 + Eq2
}
```

- Running Bernstein polynomial algorithm (equation 19) to produce knot points. The algorithm is as follows :

```
While m = 0 < total_knot, Do {
        Calculate the u knot value
        While n = 0 <  total_knot, Do {
        Calculate the v knot value
        While i = 0 < degree, Do {
                While j = 0 < degree, Do {
        Cox de Boor algorithm to knot_u
        The end control point = The first control
point * basis_value
        j = j + 1          }
        i = i + 1          }
        While  i = 0 < degree, Do {
        Cox de Boor algorithm to knot_v
        The end control point = The first control
point * basis_value
        i = i + 1          }
        Draw B-Spline surface
        n = n + 1                  }
m = m + 1                          }
```

Flowchart of the formation of B-Spline surface interpolation can be described as follows :

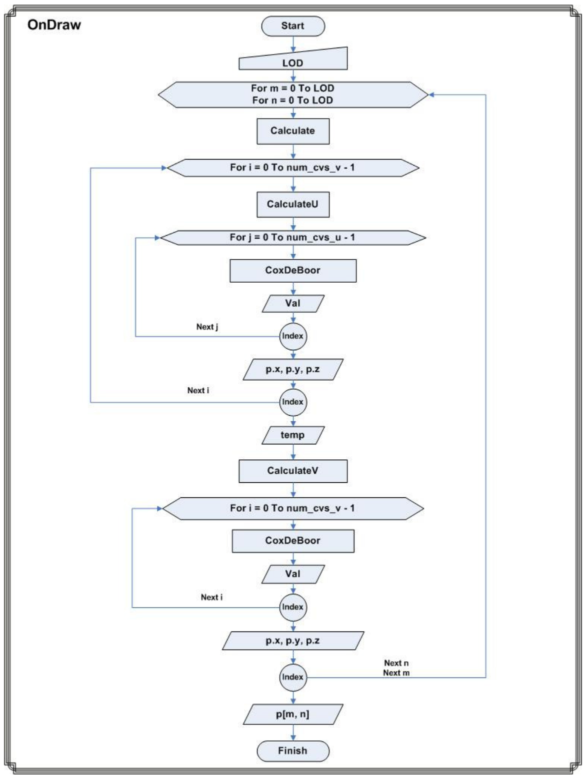


Fig. 7 Flowchart of The B-Spline Surface Interpolation

5. Implementation

Implementation of this research uses merging Visual C++ and OpenGL programming language, which the OpenGL is used to support graphic display of curves and surfaces resulting from the algorithm calculation in Visual C++.

The result of B-Spline interpolation implementation can be seen in the following pictures :

- The display of B-Spline curve

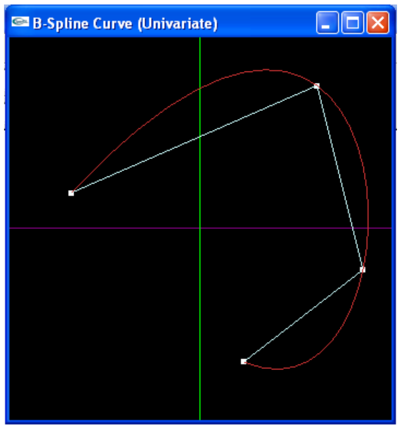
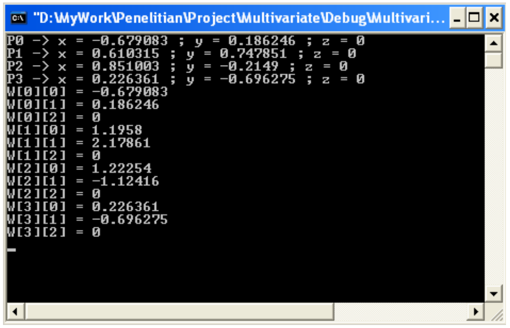


Fig. 8 Univariate Cubic B-Spline Curve with 4 Control Points



```
"D:\MyWork\Penelitian\Project\Multivariate\Debug\Multivari...
P0 -> x = -0.679083 ; y = 0.186246 ; z = 0
P1 -> x = 0.610315 ; y = 0.747851 ; z = 0
P2 -> x = 0.851003 ; y = -0.2149 ; z = 0
P3 -> x = 0.226361 ; y = -0.696275 ; z = 0
W[0][0] = -0.679083
W[0][1] = 0.186246
W[0][2] = 0
W[1][0] = 1.1958
W[1][1] = 2.17861
W[1][2] = 0
W[2][0] = 1.22254
W[2][1] = -1.12416
W[2][2] = 0
W[3][0] = 0.226361
W[3][1] = -0.696275
W[3][2] = 0
_
```

Fig. 9 The Command Prompt that Provides Control Point Coordinates (P0, P1, P2, P3) and Knot Point Matrix

The P control points in the program (figure 8) will be inputted manually (x, y, z = 0) so that it will be form a matrix which will then be multiplied by the B-Spline basis and produce the W end control points as shown in figure 9.

In figure 8, the curve arch formed very smooth and past the control points. This is because the use of 3 degree on the B-Spline interpolation and the effects of the curve itself.

Number of control points are added will also transform the B-Spline curve is formed. Generalizing the number of control points can be seen in figure 10 below :

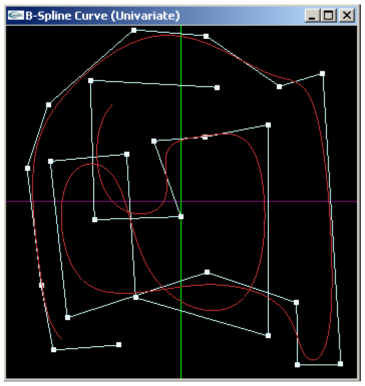


Fig. 10 Univariate Cubic B-Spline Curve with many Control Point

- The display of B-Spline surface

In this study, beside the curve also formed the Cubic B-Spline Surface. This surface on the program will be displayed in the form of control point patch with the details influenced many LOD entered.

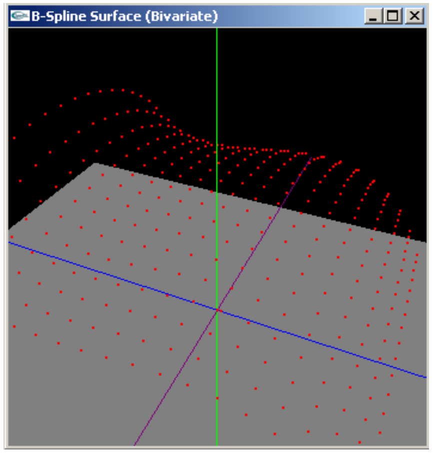
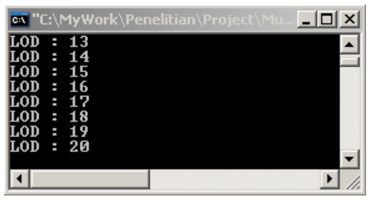


Fig. 11 Bivariate Cubic B-Spline Surface with 20 LOD



```
"C:\MyWork\Penelitian\Project\Mu...
LOD : 13
LOD : 14
LOD : 15
LOD : 16
LOD : 17
LOD : 18
LOD : 19
LOD : 20
```

Fig. 12 Command Prompt of LOD Number Addition

In figure 11 it can be seen that from 16 control points (P0 to P15), namely {10,5,10}, {5,5,10}, {-5,5,10}, {-

10,5,10}, {10,12,5}, {5,18,5}, {-5,18,5}, {-10,12,5}, {10,0,-5}, {5,0,-5}, {-5,0,-5}, {-10,0,-5}, {10,8,-10}, {5,14,-10}, {-5,14,-10}, {-10,8,-10}, the knot vector value between 0.0 to 1.0 knots and LOD as much as 20 internal knots will form a B-Spline surface with a knot point total of 2933 points. This surface curvature depends on the amount of basis value which resulting from the Cox de Boor algorithm and many degrees are used.

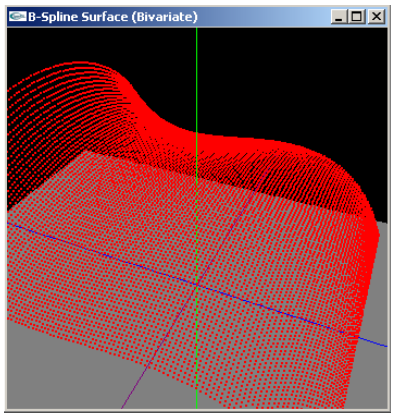


Fig. 13 Bivariate Cubic B-Spline Surface with 110 LOD

6. Conclusions

Based on the results that have been achieved in the implementation of this study, it can be concluded :

1. B-Spline interpolation method with a number of 3 degree is able to construct a curve and a surface with a smooth arch.
2. B-Spline technique is flexible due to changes in the specific control point only affects the curvature of the segment near the control point.
3. The more the number of knot (LOD) are used, the details of the B-Spline surface is formed and the more severe computational burden.
4. The interpolation method proved capable of estimating the value of curve or surface coordinate (knot) which is not known from a few control points are known

References

[1] Liliana, K. Gunadi, and F. Valiant, “Pembuatan Generator Ruled Surface dan Rotational Object dengan Menggunakan Kurva Bezier dan B-Spline”, UK Petra, Surabaya, 2006

[2] B. I Nyoman, F. Suryadi, W. Bambang, and G. Suryo, “Pemodelan B-Spline dan MARS pada Nilai Ujian Masuk terhadap IPK Mahasiswa Jurusan Desain Komunikasi Visual UK Petra Surabaya”, Jurnal Teknik Industri, Vol.8, No.1, 2006, pp. 1-13

[3] J. Ken, Geometric Modeling Notes : Definition of a B-Spline Curve, California : University of California, 1997

[4] G. Paul-Louis, and F. Stephane, Mesh Generation, London : John Wiley & Sons Inc., 2008

[5] Sangadji, Metode Numerik, Yogyakarta : Graha Ilmu, 2008

[6] M. Giuseppe, and V.M. Gradimir, Interpolation Processes : Basic Theory and Applications, Berlin Heidelberg : Springer-Verlag, 2008

[7] R. Srivastava, Polynomial Regression, New Delhi : I.A.S.R.I Library, 2004

[8] N. G. Ronald, and L. Tom, Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces, Philadelphia : SIAM Geometric Design Publication, 1993

[9] Y.Yuan and Z. Shiyu, “Sequential B-Spline Surface Construction using Multiresolution Data Clouds”, Journal of Computing and Information Science in Engineering, Vol.12, 021008-1, June 2012

[10] J. Megha, A. Pranjali, and G. Ashok, “A Journey from De-Boor’s Algorithm to De Casteljau’s Algorithm for The Construction of B-Spline Surfaces & NURBS”, International Journal of Mathematical Archive (IJMA), Vol.3, No.11, November 2012, pp. 3947-3953

[11] C. Juan, L. Xin, W. Guozhao, and Q. Hong, “Surface Reconstruction using Bivariate Simplex Splines on Delaunay Configurations”, Computers & Graphic Journal Elseiver, Vol.33, 2009, pp. 341-350

[12] H. Miklos, and J. Imre, “On Interpolation by Spline Curves with Shape Parameters”, LNCS 4975 Springer-Verlag, 2008, pp. 205-214

[13] Q. Weikang, D.R. Marc, and R. Ivo, “Uniform Approximation and Bernstein Polynomials with Coefficient in The Unit Interval”, European Journal of Combinatorics (Elseiver), Vol.32, 2011, pp. 448-463

Dian Pratiwi (27) was born in Jakarta – Indonesia on February 25, 1986. The last education is The Master of Information Technology achieved in 2011 at The University of Bina Nusantara, Jakarta – Indonesia with a GPA of 3,66. Her bachelor achieved at Trisakti University, Jakarta – Indonesia with the title of “Very Satisfied” and was awarded as “One of The 7 Best Graduate Department of Information Engineering” in 2007.
Now, the author worked as a lecturer at Trisakti University also taught courses in image processing, mobile programming, web based programming, and computer graphics began in 2008 until now. Some writings ever made that has been published is entitled “An Application of Backpropagation Artificial Neural Network Method for Measuring The Severity of Osteoarthritis”, which in 2011 published in the journal IJENS – IJET. In the journal IJCSI (Vol.9, Issue 3 No.2, May 2012), the author also made published with the title “The Use of Self Organizing Map Method and Feature Selection in Image Database Classification System”. In addition, the author also wrote another article which successfully published nationally in Indonesia at the SNTI seminar : (Jakarta, Indonesia : Trisakti University, 2010) with the title “Sistem Deteksi Penyakit Pengeroposan Tulang dengan Metode Jaringan Syaraf Tiruan Backpropagation dan Representasi Ciri dalam Ruang Eigen” and SITIA seminar (Surabaya, Indonesia : Institute of Ten November Technology – ITS, 2011) entitled “Penerapan Metode Jaringan Syaraf Tiruan Backpropagation dalam Mengukur Tingkat Keparahan Penyakit Osteoarthritis” which results in the proceedings. Now, she keep trying to develop her research on the medical by trying to apply her research interests are in Artificial Intelligence, digital image processing, and data mining.