Title:
Beta-Bezier Surfaces

Authors:
Seifalla Moustafa, seifalla.moustafa@uky.edu, University of Kentucky
Anastasia Kazadi, ansm226@g.uky.edu, University of Kentucky
Fuhua (Frank) Cheng, cheng@cs.uky.edu, University of Kentucky
Shuhua Lai, slai@ggc.edu, Georgia Gwinnett College
Alice J Lin, lina@apsu.edu, Austin Peay State University

Keywords:
Bezier curves, Bezier surfaces, Beta-Bezier Curves, Beta-Bezier Surfaces, tension control, interpolation

DOI: 10.14733/cadconfP.2023.154-158

Introduction:
When interpolating a 3D mesh, the locations of a Bezier surface's control points are determined solely by continuity conditions; that is, we cannot freely move them around. Researchers [2; 8-10] tried to find ways to extend/modify the definition of a Bezier surface so that one could reshape the surface without moving the control points, but an intuitive and straightforward approach was not available for quite a while.
Thanks to the introduction of the tension control concept into curves and triangular patches [3], [1] was able to invent a Beta-Bezier curve which (adopting the tensor-product approach) can be used to define the type of surface patches that we propose in this paper: Beta-Bezier patches. A Beta-Bezier curve segment is defined as

$$C(t;\beta) = \sum_{k=0}^{n} \mathbf{P_k} B_k^n(t;\beta) \quad ,$$

where

$$B_k^n(t;\beta) = \binom{n}{k} \frac{\prod_{i=0}^{k-1}(t+i\beta)\prod_{j=0}^{n-1-k}((1-t)+\mathrm{j}\beta)}{\prod_{m=0}^{n-1}(1+m\beta)} \quad . \qquad (1)$$

In the latter formula, n is the degree, and k is the control point index. A surface flattens out as its beta parameter increases. In addition to extending the work of [1] to surfaces, we propose an efficient rectangular mesh interpolation scheme that makes use of Beta-Bezier patches. Our scheme yields C^2-continuous piecewise Beta-Bezier surfaces which improves on the existing algorithms [5-7] in two ways:

- The existing algorithms yield G^1-continuous surfaces; ours yields C^2-continuous surfaces.
- Surfaces produced by the existing algorithms cannot be reshaped; ours can be.

Main Idea:
A Bezier surface patch is defined as the locus of a moving, deforming Bezier curve segment. An important assumption here is that each control point $\mathbf{P_i(u)}$ moves along a Bezier curve which has its own control points. The equation of a moving Bezier curve segment is

$$B(u,v) = \sum_{i=0}^{n} \binom{n}{i} v^i (1-v)^{n-i} \boldsymbol{P_i(u)}$$

Our discussion suggests that we have a grid of control points. This grid of control points is called a control net. Each control point is denoted by $\mathbf{P_{ij}}$. Since $\mathbf{P_i(u)}$ is a Bezier curve, it is defined as

$$\boldsymbol{P_i(u)} = \sum_{j=0}^{n} \binom{n}{j} u^j (1-u)^{n-j} \boldsymbol{P_{ij}}$$

If we substitute $\sum_{j=0}^{n} \binom{n}{j} u^j (1-u)^{n-j} \boldsymbol{P_{ij}}$ for $\mathbf{P_i(u)}$ in the equation of a moving Bezier curve segment, we get $\sum_{i=0}^{n} \binom{n}{i} v(1-v)^{n-i} \sum_{j=0}^{n} \binom{n}{j} u^j (1-u)^{n-j} \boldsymbol{P_{ij}} = \sum_{i=0}^{n} \sum_{j=0}^{n} \binom{n}{i} \binom{n}{j} u^j (1-u)^{n-j} v(1-v)^{n-i} \boldsymbol{P_{ij}} = \sum_{i=0}^{n} \sum_{j=0}^{n} B_j^n(u) B_i^n(v) \boldsymbol{P_{ij}}$,
which is the equation of a Bezier surface patch. A Beta-Bezier surface patch is defined as the locus of a moving, deforming Beta-Bezier curve segment. Proceeding in the same manner as above, we get

$$B(u,v;\beta_u;\beta_v) = \sum_{i=0}^{n} \sum_{j=0}^{n} B_j^n(u;\beta_u) B_i^n(v;\beta_v) \boldsymbol{P_{ij}},$$

which is the equation of a Beta-Bezier surface ($B_i^n(v;\beta_v)$ and $B_j^n(u;\beta_u)$ are defined by equation (1)).

A bicubic Beta-Bezier surface patch can be represented by a bicubic Bezier surface patch. Recall that a Beta-Bezier surface patch is defined as the locus of a moving, deforming Beta-Bezier curve segment. So, it can be written as $B(u,v;\beta_u;\beta_v) = \sum_{i=0}^{n} B_i^n(v;\beta_v) \boldsymbol{P_i(u;\beta_u)}$ **(2)**. For each $\boldsymbol{P_i(u;\beta_u)}$ we find $\bar{P}_{0,j}, \bar{P}_{1,j}, \bar{P}_{2,j}$ and $\bar{P}_{3,j}$, so that $\boldsymbol{P_i(u;\beta_u)}$ can be expressed as a cubic Bezier curve in u as follows:

$$\boldsymbol{P_i(u;\beta_u)} = \sum_{i=0}^{3} \bar{P}_{i,j} B_i^3(u) \tag{3}$$

$\bar{P}_{0,j}, \bar{P}_{1,j}, \bar{P}_{2,j}$ and $\bar{P}_{3,j}$ can be found as follows:

$$\bar{P}_{0,j} = P_{0,j}$$

$$\bar{P}_{1,j} = \frac{\beta(3+4\beta)}{3(1+\beta)(1+2\beta)} P_{0,j} + \frac{1}{1+2\beta} P_{1,j} + \frac{\beta}{(1+\beta)(1+2\beta)} P_{2,j} + \frac{2\beta^2}{3(1+\beta)(1+2\beta)} P_{3,j}$$

$$\bar{P}_{2,j} = \frac{2\beta^2}{3(1+\beta)(1+2\beta)} P_{0,j} + \frac{\beta}{(1+\beta)(1+2\beta)} P_{1,j} + \frac{1}{1+2\beta} P_{2,j} + \frac{\beta(3+4\beta)}{3(1+\beta)(1+2\beta)} P_{3,j}$$

$$\bar{P}_{3,j} = P_{3,j}$$

By substituting (3) into (2), we have

$$\sum_{j=0}^{3} [\sum_{i=0}^{3} \bar{P}_{i,j} B_i^3(u)] B_j^3(v;\beta_v)$$

$$\sum_{i=0}^{3} [\sum_{j=0}^{3} \bar{P}_{i,j} B_j^3(v;\beta_v)] B_i^3(u) = \sum_{i=0}^{3} \bar{P}_i(v;\beta_v) B_i^3(u) \tag{4}$$

For each $\bar{P}_i(v;\beta)$, we find $Q_{i,0}, Q_{i,1}, Q_{i,2}$ and $Q_{i,3}$ using the same procedure as above, so that $\bar{P}_i(v;\beta)$ can be expressed as a cubic Bezier curve in v as follows:

$$\sum_{i=0}^{3} Q_{i,j} B_{3,j}(v) \tag{5}$$

By substituting (5) into (4), we have a bicubic Bezier surface patch representation for $B(u,v;\beta_u;\beta_v)$. Figures 1 and 2 show two examples of Beta-Bezier surface patches.

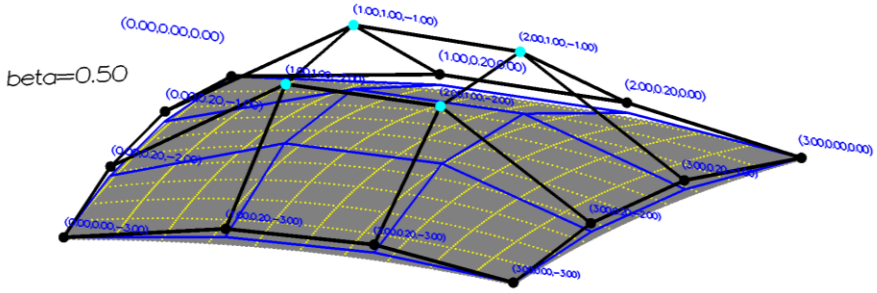

Fig. 1: The blue control net is the control net for the Bezier surface patch, the black one is for the Beta-Bezier surface patch.

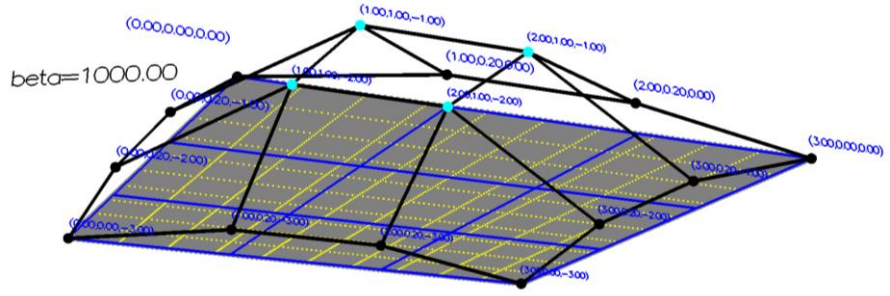

Fig. 2: The blue control net is the control net for the Bezier surface patch, the black one is for the Beta-Bezier surface patch.

As can be seen from the figures, the surface flattens out as Beta increases. It should be emphasized that the Beta-Bezier control points did not have to change to get from figure 1 to figure 2; the only difference between the two figures is the value of Beta.

A Beta-Bezier surface patch interpolates its four corner control points. This means that complex shapes can be modeled using a composite Bezier surface (i.e. multiple patches pieced together). [3] For any two patches to meet smoothly (e.g. to be C^n continuous), there are two conditions that must be met:

- The two patches must have a common boundary
- All the columns of their control nets (or rows if the two patches are to be pieced together sideways) must be control polygons for C^n continuous curves.

Let's suppose that there is a 3D mesh that we wish to interpolate. Our algorithm works as follows. It (a) generates longitudinal curves interpolating the columns of the data mesh in question, (b) generates latitudinal curves interpolating the rows of the data mesh (including the rows generated by step (a)), and finally (c) generates a piecewise surface using the control points computed in the process. 3D meshes can be represented in a variety of ways, none of which explicitly tells what data points comprise a column (or a row). One way to solve this problem is to use an adjacency matrix to represent the mesh and perform a depth-first search with no backtracking and a criterion for choosing the next vertex. Before we state the criterion, we should make clear that each vertex has four neighbors, one of which is the vertex we have come from, so really, we have 3 neighbors to choose from. Here is how we choose: We calculate the angle between the edge that connects each neighbor with the current vertex and the edge between the current vertex and the previous one, and then we take the median of those angles. Composite Beta-Bezier surfaces are shown in figures 3, 4, and 5.

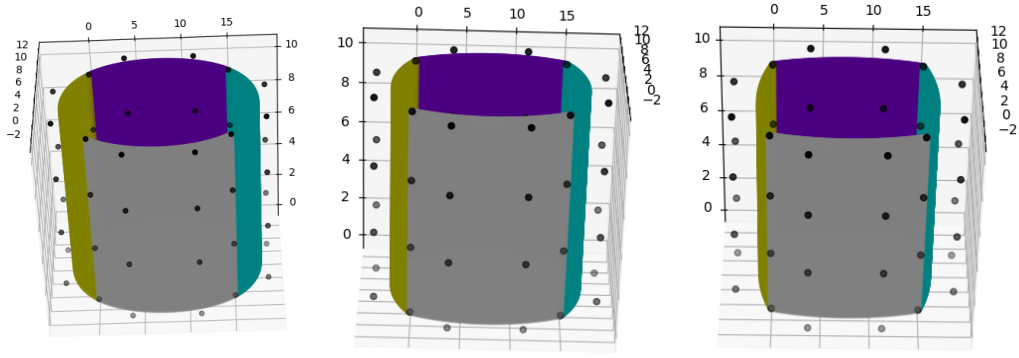

Fig. 3: Composite surface with Beta = 0, Beta = 0.5, and Beta = 1.5.

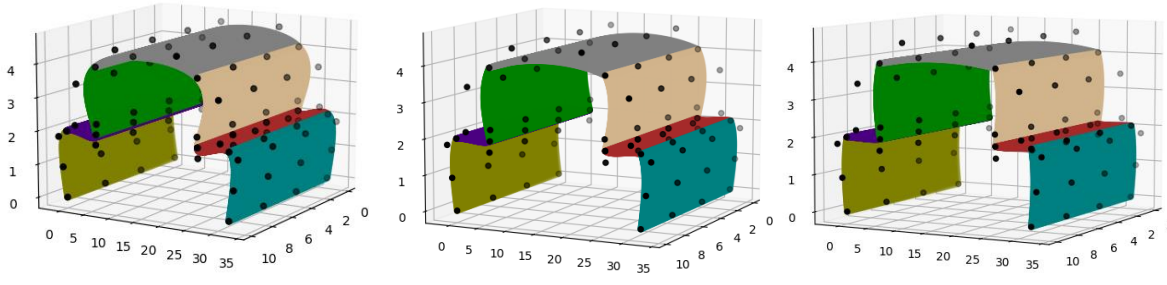

Fig. 4: Composite surface with Beta = 0, Beta = 0.5, and Beta = 1.5.

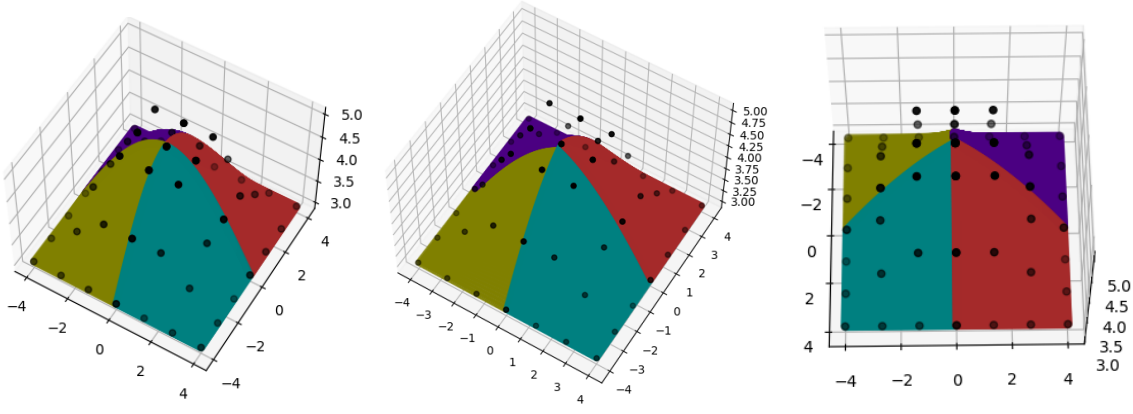

Fig. 5: Composite surface with Beta = 0, Beta = 0.5, and Beta = 1.5.

To prove the correctness of our algorithm, we need to prove that it produces the right number of control points for each patch and that the control points meet the continuity conditions. We start with the former. We will prove it for the bicubic case. To generate a bicubic Beta-Bezier patch, we need 16

control points. If we interpolate a column of the data mesh, we get two control points for every segment. After interpolating all the columns, each patch has 4 control points plus the 4 data points, for a total of 8 control points. Put another way, each patch has four rows of control points each containing 2 control points. If we interpolate those rows, we get 8 additional control points, totaling 16 control points. Curve interpolation, as described in [1], generates cubic curves that are C^2 continuous. Therefore, the columns (and rows) of the control nets of the adjacent patches produced by our algorithm are control polygons of C^2 continuous Beta-Bezier curves (i.e., the second continuity condition). We start by interpolating the data points. This guarantees that we have common boundary curves (i.e., the first continuity condition), which completes our proof.

As for the running time of our algorithm, let's suppose there are n columns and m rows on the mesh in question. As explained in [1], interpolating m points involves solving an 2mx2m system which, in a worst-case scenario, takes $O(m^3)$ time. Therefore, step (a) of our algorithm takes $O(nm^3)$ time. Following similar reasoning, we can conclude that step (b) takes $O(mn^3)$ time, for a total running time of $O(nm^3+mn^3)$.

Conclusion:

In this paper, we extend the concept of tension control proposed in [1] from curves to surfaces. The proposed type of surface patches can be reshaped without moving its control points. In addition to extending the work of [1] to surfaces, we propose an efficient rectangular mesh interpolation scheme that makes use of the proposed type of patches.

One thing that should be studied is how a Beta-Bezier surface can be represented as a B-spline surface. Also, our interpolation scheme only works with rectangular grids. Meshes with arbitrary topology should also be considered. More work is needed to address the above.

References:

[1] Cheng, F.; Kazadi, A.; Lin, A.: Beta-Bezier curves. CAD'20, 2020, https://doi.org/10.14733/cadconfp.2020.343-347

[2] Cao, J.; Wang, G.Z.: An extension of Bernstein-Bezier surface over the triangular domain, Progress Nat. Sci. 17, 2007, 352-357. https://doi.org/10.1080/10020070612331343269

[3] Chu, L.; Zeng, X.M.: Constructing curves and triangular patches by Beta functions. Journal of Computational and Applied Mathematics, 260, 2014, 191-200. https://doi.org/10.1016/j.cam.2013.09.025

[4] Farin, G.E.: Curves and surfaces for computer aided geometric design: A practical guide. Academic Press, 1988. https://doi.org/10.1016/B978-0-12-460515-2.50020-2

[5] Lin, H.; Chen, W.; Bao, H.: Adaptive patch-based mesh fitting for reverse engineering. In Computer-Aided Design, 39(12), 2007, 1134–1142, Elsevier BV. https://doi.org/10.1016/j.cad.2007.10.002

[6] Eck, M.; Hoppe, H.: Automatic reconstruction of B-spline surfaces of arbitrary topological type. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). Assoc. for Computing Machinery, New York, NY, USA, 1996, 325–334. https://doi.org/10.1145/237170.237271

[7] Shirman L.; Sequin C.: Local surface interpolation with Bezier patches, Computer Aided Geometric Design, 4, 1987, 279-95. https://doi.org/10.1016/0167-8396(87)90003-3

[8] Yan, L.L.; Liang, J.F.: An extension of the Bezier model, Applied Mathematics and Computation 218, 2011, 2863-2879. https://doi.org/10.1016/j.amc.2011.08.030

[9] Yang, L.Q.; Zeng, X.M.: Bezier curves and surfaces with shape parameters. Int. J. Comput. Math. 86, 2009, 1253-1263. https://doi.org/10.1080/00207160701821715

[10] Zhu, Y.; Han, X.: Quasi-Bernstein-Bezier polynomials over triangular domain with multiple shape parameters, Applied Mathematics and Computation 250, 2015, 181-192. https://doi.org/10.1016/j.amc.2014.10.098