Parametric Surfaces

Michael Kazhdan

(601.457/657)

HB 10.6 -- 10.9, 10.13
FvDFH 11.2
Announcement

• OpenGL review session:
  ◦ When: Wednesday (11/8) @ 9:00 PM
  ◦ Where: Malone 228
Cubic Splines

Given a collection of control points, \( \{p_0, \ldots, p_n\} \), we define a collection of cubic polynomial functions \( \{P_1(u), \ldots, P_{n-2}(u)\} \) that jointly describe a curve approximating/interpolating the control points.

Each cubic function \( P_k(u) \) is defined in the region \( 0 \leq u \leq 1 \) and is determined by the four points \( p_{k-1}, p_k, p_{k+1}, \) and \( p_{k+2} \).
Cubic Blending Functions

Blending functions provide a way for expressing the functions $P_k(u)$ as a weighted sum of the four control points $p_{k-1}, p_k, p_{k+1},$ and $p_{k+2}$:

$$P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2}$$
Cubic Blending Functions

• Translation Commutativity:
  - \( BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1 \), for all \( 0 \leq u \leq 1 \).

\[
P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2}
\]
Cubic Blending Functions

• Translation Commutativity:
  \[ BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1, \text{ for all } 0 \leq u \leq 1. \]

• \( n \)-th Order Continuity:
  \[ BF_0^{(n)}(1) = BF_3^{(n)}(0) = 0 \]
  \[ BF_1^{(n)}(1) = BF_0^{(n)}(0) \]
  \[ BF_2^{(n)}(1) = BF_1^{(n)}(0) \]
  \[ BF_3^{(n)}(1) = BF_2^{(n)}(0) \]

\[ P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2} \]
Cubic Blending Functions

- Translation Commutativity:
  - \( BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1 \), for all \( 0 \leq u \leq 1 \).

- \( n \)-th Order Continuity:
  - \( BF_0^{(n)}(1) = BF_3^{(n)}(0) = 0 \)
  - \( BF_1^{(n)}(1) = BF_0^{(n)}(0) \)
  - \( BF_2^{(n)}(1) = BF_1^{(n)}(0) \)
  - \( BF_3^{(n)}(1) = BF_2^{(n)}(0) \)

- Hull Containment:
  - \( BF_0(u), BF_1(u), BF_2(u), BF_3(u) \leq 1 \), for all \( 0 \leq u \leq 1 \).

\[ P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2} \]
Cubic Blending Functions

- **Translation Commutativity:**
  - \( BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1 \), for all \( 0 \leq u \leq 1 \).

- **n-th Order Continuity:**
  - \( BF_0^{(n)}(1) = BF_3^{(n)}(0) = 0 \)
  - \( BF_1^{(n)}(1) = BF_0^{(n)}(0) \)
  - \( BF_2^{(n)}(1) = BF_1^{(n)}(0) \)
  - \( BF_3^{(n)}(1) = BF_2^{(n)}(0) \)

- **Hull Containment:**
  - \( BF_0(u), BF_1(u), BF_2(u), BF_3(u) \leq 1 \), for all \( 0 \leq u \leq 1 \).

- **Interpolation:**
  - \( BF_0(0) = BF_2(0) = BF_3(0) = 0 \)
  - \( BF_0(1) = BF_1(1) = BF_3(1) = 0 \)
  - \( BF_1(0) = 1 \)
  - \( BF_2(1) = 1 \)

\[
P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2}
\]
Overview

From Curves to surfaces

- Spline Curves and Blending Functions
- Weighted Averaging
- Spline Surfaces
- Spline Surface Properties
Weighted Averaging

Suppose we have an array of values:

- \( v_1, v_2, v_3, \) and \( v_4, \)

and we have weights:

- \( \alpha_1, \alpha_2, \alpha_3, \) and \( \alpha_4, \) with \( \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1, \)
- \( \beta_1, \beta_2, \beta_3, \) and \( \beta_4, \) with \( \beta_1 + \beta_2 + \beta_3 + \beta_4 = 1. \)

We can express the weighted average of the \( v_i \) in matrix form:

\[
\sum_{i=1}^{4} \alpha_i v_i = (\alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4) \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \quad \sum_{i=1}^{4} \beta_i v_i = (v_1 \ v_2 \ v_3 \ v_4) \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix}
\]
Weighted Averaging

If we have a matrix of values:

\[
\begin{pmatrix}
v_{11} & v_{21} & v_{31} & v_{41} \\
v_{12} & v_{22} & v_{32} & v_{42} \\
v_{13} & v_{23} & v_{33} & v_{43} \\
v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\]

multiplying on the left by \((\alpha_1 \alpha_2 \alpha_3 \alpha_4)\) gives:

\[
(\alpha_1 \alpha_2 \alpha_3 \alpha_4) \begin{pmatrix}
v_{11} & v_{21} & v_{31} & v_{41} \\
v_{12} & v_{22} & v_{32} & v_{42} \\
v_{13} & v_{23} & v_{33} & v_{43} \\
v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\]
Weighted Averaging

If we have a matrix of values:

\[
\begin{pmatrix}
v_{11} & v_{21} & v_{31} & v_{41} \\
v_{12} & v_{22} & v_{32} & v_{42} \\
v_{13} & v_{23} & v_{33} & v_{43} \\
v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\]

multiplying on the left by \((\alpha_1 \: \alpha_2 \: \alpha_3 \: \alpha_4)\) gives:

\[
\begin{pmatrix}
\alpha_1 \\
\alpha_2 \\
\alpha_3 \\
\alpha_4
\end{pmatrix}
\begin{pmatrix}
v_{11} & v_{21} & v_{31} & v_{41} \\
v_{12} & v_{22} & v_{32} & v_{42} \\
v_{13} & v_{23} & v_{33} & v_{43} \\
v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\]

\[=\]

\[
\begin{pmatrix}
\sum \alpha_i v_{1i} \\
\sum \alpha_i v_{2i} \\
\sum \alpha_i v_{3i} \\
\sum \alpha_i v_{4i}
\end{pmatrix}
\]

\[\Rightarrow\text{ A row vector with entries that are the weighted average of the columns of the matrix.}\]
Weighted Averaging

Similarly, if we have a matrix of values:

\[
\begin{pmatrix}
  v_{11} & v_{21} & v_{31} & v_{41} \\
  v_{12} & v_{22} & v_{32} & v_{42} \\
  v_{13} & v_{23} & v_{33} & v_{43} \\
  v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\]

multiplying on the right by \((\beta_1 \beta_2 \beta_3 \beta_4)^t\) gives:

\[
\begin{pmatrix}
  v_{11} & v_{21} & v_{31} & v_{41} \\
  v_{12} & v_{22} & v_{32} & v_{42} \\
  v_{13} & v_{23} & v_{33} & v_{43} \\
  v_{14} & v_{24} & v_{34} & v_{44}
\end{pmatrix}
\begin{pmatrix}
  \beta_1 \\
  \beta_2 \\
  \beta_3 \\
  \beta_4
\end{pmatrix} =
\begin{pmatrix}
  \sum \beta_j v_{j1} \\
  \sum \beta_j v_{j2} \\
  \sum \beta_j v_{j3} \\
  \sum \beta_j v_{j4}
\end{pmatrix}
\]

\Rightarrow \text{A column vector with entries that are the weighted average of the rows of the matrix.}
Weighted Averaging

Simultaneously multiplying on the left by \((\alpha_1 \alpha_2 \alpha_3 \alpha_4)\) and on the right by \((\beta_1 \beta_2 \beta_3 \beta_4)^t\) gives:

\[
(\alpha_1 \alpha_2 \alpha_3 \alpha_4)\begin{pmatrix} v_{11} & v_{21} & v_{31} & v_{41} \\ v_{12} & v_{22} & v_{32} & v_{42} \\ v_{13} & v_{23} & v_{33} & v_{43} \\ v_{14} & v_{24} & v_{34} & v_{44} \end{pmatrix}\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} = \mathbf{i}j,
Weighted Averaging

Simultaneously multiplying on the left by \((\alpha_1 \alpha_2 \alpha_3 \alpha_4)\) and on the right by \((\beta_1 \beta_2 \beta_3 \beta_4)^t\) gives:

\[
\begin{pmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\end{pmatrix}
\begin{pmatrix}
v_{11} & v_{21} & v_{31} & v_{41} \\
v_{12} & v_{22} & v_{32} & v_{42} \\
v_{13} & v_{23} & v_{33} & v_{43} \\
v_{14} & v_{24} & v_{34} & v_{44} \\
\end{pmatrix}
\begin{pmatrix}
\beta_1 \\
\beta_2 \\
\beta_3 \\
\beta_4 \\
\end{pmatrix}
= \sum_{i,j=1}^{4} \alpha_i \beta_j v_{ij}
\]

⇒ The weighted sum of the \(v_{ij}\), weighted by \(\alpha_j \beta_i\).

This is a weighted average of the \(v_{ij}\):

To show this, we have to show that the total sum of the weights \(\alpha_i \beta_j\) is equal to 1.
**Weighted Averaging**

Simultaneously multiplying on the left by \((\alpha_1 \alpha_2 \alpha_3 \alpha_4)\) and on the right by \((\beta_1 \beta_2 \beta_3 \beta_4)^t\) gives:

\[
(\alpha_1 \alpha_2 \alpha_3 \alpha_4) \begin{pmatrix} v_{11} & v_{21} & v_{31} & v_{41} \\ v_{12} & v_{22} & v_{32} & v_{42} \\ v_{13} & v_{23} & v_{33} & v_{43} \\ v_{14} & v_{24} & v_{34} & v_{44} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} = \sum_{i,j=1}^{4} \alpha_i \beta_j v_{ij}
\]

⇒ The weighted sum of the \(v_{ij}\), weighted by \(\alpha_j \beta_i\).

This is a weighted **average** of the \(v_{ij}\):

\[
\sum_{i,j=1}^{4} \alpha_i \beta_j = \sum_{i=1}^{4} \alpha_i \left( \sum_{j=1}^{4} \beta_j \right) = \sum_{i=1}^{4} \alpha_i = 1
\]
Overview

From Curves to surfaces

• Spline Curves and Blending Functions
• Weighted Averaging
• Spline Surfaces
• Spline Surface Properties
Spline Surfaces

• A parametric curve is a function in one variable $\Phi(u)$ associating a position to every value of $u$. 

\[
\Phi(0) \rightarrow \Phi(1/8) \rightarrow \Phi(1/4) \rightarrow \Phi(3/8) \rightarrow \Phi(1/2) \rightarrow \Phi(3/4) \rightarrow \Phi(5/8) \rightarrow \Phi(7/8) \rightarrow \Phi(1)
\]
Spline Surfaces

- A **parametric curve** is a function in one variable $\Phi(u)$ associating a position to every value of $u$.
- A **parametric patch** is a function in two variables $\Phi(u,v)$ that associates a position to every pair of values of $(u,v)$. 
Spline Surfaces

- When considering spline curves, we use four control points to define a cubic polynomial $P_k(u)$ in one variable ($0 \leq u \leq 1$).
Spline Surfaces

• When considering spline curves, we use four control points to define a cubic polynomial $P_k(u)$ in one variable ($0 \leq u \leq 1$).

• When considering spline surfaces, we use $4 \times 4$ control points to define a bi-cubic polynomial $P_{k,l}(u, v)$ in two variables ($0 \leq u, v \leq 1$).
Spline Surfaces

• When considering spline curves, we use four control points to define a cubic polynomial $P_k(u)$ in one variable ($0 \leq u \leq 1$).

• When considering spline surfaces, we use $4 \times 4$ control points to define a bi-cubic polynomial $P_{k,l}(u, v)$ in two variables ($0 \leq u, v \leq 1$).

A bi-cubic polynomial is a polynomial which is cubic in each variable:

$$P(u, v) = au^3v^3 + 
+ bu^3v^2 + cu^2v^3 + 
+ du^2v^2 + eu^1v^3 + fu^3v^1 + 
+ \ldots$$
Spline Surfaces

- Given \( n \) points, we generate a piecewise cubic curve consisting of \( n - 3 \) segments that approximate/interpolate the points.
Spline Surfaces

• Given $n$ points, we generate a piecewise cubic curve consisting of $n - 3$ segments that approximate/interpolate the points.

• Given $n \times m$ points, we generate a piecewise bicubic curve, consisting of $(n - 3) \times (m - 3)$ patches that approximate/interpolate the points.
Spline Surfaces

We generate spline curves by using the blending function to compute the weighted average of the control points.

We do the same for surfaces.
Cubic Blending Functions

Recall:

For a cubic segment of a spline curve, we can express the spline curve in matrix form as:

\[
P_k(u) = \begin{pmatrix}
    BF_0(u) \\
    BF_1(u) \\
    BF_2(u) \\
    BF_3(u)
\end{pmatrix}^t
\begin{pmatrix}
    p_{k-1} \\
    p_k \\
    p_{k+1} \\
    p_{k+2}
\end{pmatrix}
\]

Since the sum of the \( BF_i(u) \) is always equal to 1, this is a weighted average of the control points.

\[
P_k(u) = BF_0(u)p_{k-1} + BF_1(u)p_k + BF_2(u)p_{k+1} + BF_3(u)p_{k+2}
\]
Cubic Blending Functions

If we are given a $4 \times 4$ array of control points, we can define a bi-cubic spline patch similarly:

$$P_{k,l}(u, v) = \begin{pmatrix} BF_0(u) \\ BF_1(u) \\ BF_2(u) \\ BF_3(u) \end{pmatrix}^t \begin{pmatrix} p_{k-1,l-1} & p_{k,l-1} & p_{k+1,l-1} & p_{k+2,l-1} \\ p_{k-1,l} & p_{k,l} & p_{k+1,l} & p_{k+2,l} \\ p_{k-1,l+1} & p_{k,l+1} & p_{k+1,l+1} & p_{k+2,l+1} \\ p_{k-1,l+2} & p_{k,l+2} & p_{k+1,l+2} & p_{k+2,l+2} \end{pmatrix} \begin{pmatrix} BF_0(v) \\ BF_1(v) \\ BF_2(v) \\ BF_3(v) \end{pmatrix}$$

Again, the sum of the $BF_i(u)$ equals 1, so $P_{k,l}(u, v)$ is a weighted average of the control points.
Cubic Spline Patches

We can choose our favorite spline curve (Cardinal, Catmull-Rom, uniform cubic-B, etc.) and use its blending functions to define a spline patch:

\[
P_{k,l}(u, v) = \begin{pmatrix} BF_0(u) \\ BF_1(u) \\ BF_2(u) \\ BF_3(u) \end{pmatrix}^t \begin{pmatrix} \begin{pmatrix} p_{k-1,l-1} & p_{k,l-1} & p_{k+1,l-1} & p_{k+2,l-1} \\ p_{k-1,l} & p_{k,l} & p_{k+1,l} & p_{k+2,l} \\ p_{k-1,l+1} & p_{k,l+1} & p_{k+1,l+1} & p_{k+2,l+1} \\ p_{k-1,l+2} & p_{k,l+2} & p_{k+1,l+2} & p_{k+2,l+2} \end{pmatrix} \end{pmatrix} \begin{pmatrix} BF_0(v) \\ BF_1(v) \\ BF_2(v) \\ BF_3(v) \end{pmatrix}
\]
Computing the value of the patch at some point \((u_0, v_0)\) amounts to:

1. Averaging the rows using the weights \(BF_i(v_0)\)
2. Averaging the result using the weights \(BF_i(u_0)\).

\[
P_{k,l}(u, v) = \begin{pmatrix} BF_0(u) \\ BF_1(u) \\ BF_2(u) \\ BF_3(u) \end{pmatrix}^t \begin{pmatrix} p_{k-1,l-1} & p_{k,l-1} & p_{k+1,l-1} & p_{k+2,l-1} \\ p_{k-1,l} & p_{k,l} & p_{k+1,l} & p_{k+2,l} \\ p_{k-1,l+1} & p_{k,l+1} & p_{k+1,l+1} & p_{k+2,l+1} \\ p_{k-1,l+2} & p_{k,l+2} & p_{k+1,l+2} & p_{k+2,l+2} \end{pmatrix} \begin{pmatrix} BF_0(v) \\ BF_1(v) \\ BF_2(v) \\ BF_3(v) \end{pmatrix}
\]
Cubic Spline Patches

Computing the value of the patch at some point \((u_0, v_0)\) amounts to:

1. Averaging the rows using the weights \(BF_i(v_0)\)
2. Averaging the result using the weights \(BF_i(u_0)\).
Cubic Spline Patches

Expanding the matrix we get:

\[ P_{k,l}(u, v) = BF_0(u) \cdot BF_0(v) \cdot p_{k-1,l-1} + BF_0(u) \cdot BF_1(v) \cdot p_{k-1,l} + \cdots \]
\[ + BF_1(u) \cdot BF_0(v) \cdot p_{k,l-1} + BF_1(u) \cdot BF_1(v) \cdot p_{k,l} + \cdots \]
\[ + \cdots \]

Or, if we set \( BF_{i,j}(u, v) = BF_i(u) \cdot BF_j(v) \) we get:

\[ P_{k,l}(u, v) = BF_{0,0}(u, v) \cdot p_{k-1,l-1} + BF_{0,1}(u, v) \cdot p_{k-1,l} + \cdots \]
\[ + BF_{1,0}(u, v) \cdot p_{k,l-1} + BF_{1,1}(u, v) \cdot p_{k,l} + \cdots \]
\[ + \cdots \]
Cubic Spline Patches

Alternatively, we can use the matrix formulation:

\[
\begin{pmatrix}
BF_0(u) \\
BF_1(u) \\
BF_2(u) \\
BF_3(u)
\end{pmatrix}^t = UM_{\text{Spline}}
\]

with \( U = (u^3 \ u^2 \ u \ 1) \) and \( M_{\text{Spline}} \) the spline matrix.

This gives:

\[
P_{k,l}(u, v) = UM_{\text{Spline}}\begin{pmatrix}
p_{k-1,l-1} & p_{k,l-1} & p_{k+1,l-1} & p_{k+2,l-1} \\
p_{k-1,l} & p_{k,l} & p_{k+1,l} & p_{k+2,l} \\
p_{k-1,l+1} & p_{k,l+1} & p_{k+1,l+1} & p_{k+2,l+1} \\
p_{k-1,l+2} & p_{k,l+2} & p_{k+1,l+2} & p_{k+2,l+2}
\end{pmatrix}M_{\text{Spline}}^tV^t
\]

with \( V = (v^3 \ v^2 \ v \ 1) \).
Cubic Spline Patches

Alternatively, we can use the matrix formulation:

\[
\begin{pmatrix}
BF_0(u) \\
BF_1(u) \\
BF_2(u) \\
BF_3(u)
\end{pmatrix}^t = U M_{\text{Spline}}
\]

Surface splines that are obtained from curve splines in this way are often referred to as tensor product splines.

This gives:

\[
P_{k,l}(u, v) = U M_{\text{Spline}} \begin{pmatrix}
p_{k-1,l-1} & p_{k,l-1} & p_{k+1,l-1} & p_{k+2,l-1} \\
p_{k-1,l} & p_{k,l} & p_{k+1,l} & p_{k+2,l} \\
p_{k-1,l+1} & p_{k,l+1} & p_{k+1,l+1} & p_{k+2,l+1} \\
p_{k-1,l+2} & p_{k,l+2} & p_{k+1,l+2} & p_{k+2,l+2}
\end{pmatrix} M_{\text{Spline}}^t V^t
\]

with \( V = (v^3 \, v^2 \, v \, 1) \).
Overview

From Curves to surfaces

• Spline Curves and Blending Functions
• Weighted Averaging
• Spline Surfaces
• Spline Surface Properties
Surface Spline Properties

We began by describing some of the properties that we would like spline curves to satisfy:

- **Translation commutativity:**
  \[ BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1, \text{ for all } 0 \leq u \leq 1. \]

- **n-th order continuity:**
  \[ BF_0^{(n)}(1) = BF_3^{(n)}(0) = 0 \]
  \[ BF_1^{(n)}(1) = BF_0^{(n)}(0) \]
  \[ BF_2^{(n)}(1) = BF_1^{(n)}(0) \]
  \[ BF_3^{(n)}(1) = BF_2^{(n)}(0) \]

- **Hull containment:**
  \[ BF_0(u), BF_1(u), BF_2(u), BF_3(u) \leq 1, \text{ for all } 0 \leq u \leq 1. \]

- **Interpolation:**
  \[ BF_0(0) = BF_2(0) = BF_3(0) = 0 \]
  \[ BF_0(1) = BF_1(1) = BF_3(1) = 0 \]
  \[ BF_1(0) = 1 \]
  \[ BF_2(1) = 1 \]
Surface Spline Properties

We began by describing some of the properties that we would like spline curves to satisfy:

- **Translation commutativity:**
  - \( BF_0(u) + BF_1(u) + BF_2(u) + BF_3(u) = 1 \), for all \( 0 \leq u \leq 1 \).

- **\( n \)-th order continuity:**
  - \( BF_0^{(n)}(1) = BF_3^{(n)}(0) = 0 \)
  - \( BF_1^{(n)}(1) = BF_0^{(n)}(0) \)
  - \( BF_2^{(n)}(1) = BF_1^{(n)}(0) \)
  - \( BF_3^{(n)}(1) = BF_2^{(n)}(0) \)

- **Hull containment:**
  - \( BF_0(u), BF_1(u), BF_2(u), BF_3(u) \leq 1 \), for all \( 0 \leq u \leq 1 \).

- **Interpolation:**
  - \( BF_0(0) = BF_2(0) = BF_3(0) = 0 \)
  - \( BF_0(1) = BF_1(1) = BF_3(1) = 0 \)
  - \( BF_1(0) = 1 \)

- Do the tensor product splines satisfy these conditions?
Surface Spline Properties

• **Translation commutativity:**
  - We have shown that if the tensor spline is generated from weighting function that sum to 1, then the tensor product weighting functions will also sum to 1.
Surface Spline Properties

• **Continuity:**
  
  It turns out that the conditions for a tensor spline to be continuous are:

  » \( BF_{0,i}(1, \nu) = BF_{i,0}(u, 1) = 0 \)
  » \( BF_{3,i}(0, \nu) = BF_{i,3}(u, 0) = 0 \)
  » \( BF_{1,i}(1, \nu) = BF_{0,i}(0, \nu) \)
  » \( BF_{i,1}(u, 1) = BF_{i,0}(u, 0) \)
  » \( BF_{2,i}(1, \nu) = BF_{1,i}(0, \nu) \)
  » \( BF_{i,2}(u, 1) = BF_{i,1}(u, 0) \)
  » \( BF_{3,i}(1, \nu) = BF_{2,i}(0, \nu) \)
  » \( BF_{i,3}(u, 1) = BF_{i,2}(u, 0) \)
Surface Spline Properties

• **Continuity:**
  - It turns out that the conditions for a tensor spline to be continuous are:

\[
\begin{align*}
  BF_{0,i}(1, v) &= BF_{i,0}(u, 1) = 0 \\
  BF_{3,i}(0, v) &= BF_{i,3}(u, 0) = 0 \\
  BF_{1,i}(1, v) &= BF_{0,i}(0, v) \\
  BF_{i,1}(u, 1) &= BF_{i,0}(u, 0) \\
  BF_{2,i}(1, v) &= BF_{1,i}(0, v) \\
  BF_{i,2}(u, 1) &= BF_{i,1}(u, 0) \\
  BF_{3,i}(1, v) &= BF_{2,i}(0, v) \\
  BF_{i,3}(u, 1) &= BF_{i,2}(u, 0)
\end{align*}
\]
Surface Spline Properties

- **Continuity:**
  - It turns out that the conditions for a tensor spline to be continuous are:
    
    $BF_{0,i}(1,v) = BF_{i,0}(u,1) = 0$
    $BF_{3,i}(0,v) = BF_{i,3}(u,0) = 0$
    $BF_{1,i}(1,v) = BF_{0,i}(0,v)$
    $BF_{i,1}(u,1) = BF_{i,0}(u,0)$
    $BF_{2,i}(1,v) = BF_{1,i}(0,v)$
    $BF_{i,2}(u,1) = BF_{i,1}(u,0)$
    $BF_{3,i}(1,v) = BF_{2,i}(0,v)$
    $BF_{i,3}(u,1) = BF_{i,2}(u,0)$
Surface Spline Properties

- **Continuity:**
  - It turns out that the conditions for a tensor spline to be continuous are:

    » $BF_{0,i}(1, v) = BF_{i,0}(u, 1) = 0$
    
    » $BF_{3,i}(0, v) = BF_{i,3}(u, 0)$
    
    » $BF_{1,i}(1, v) = BF_{i,1}(u, 0)$
    
    » $BF_{2,i}(1, v) = BF_{i,2}(u, 0)$
    
    » $BF_{3,i}(1, v) = BF_{2,i}(0, v)$
    
    » $BF_{i,3}(u, 1) = BF_{i,2}(u, 0)$

    If $BF_0(1) = 0$, then since:
    
    $$BF_{0,i}(u, v) = BF_0(u) \cdot BF_i(v)$$

    we must have $BF_{0,i}(1, v) = 0$ for all $i$ and all $v$. 
Surface Spline Properties

• Continuity:
  - It turns out that the conditions for a tensor spline to be continuous are:
    - \( BF_{0,i}(1, v) = BF_{i,0}(u, 1) = 0 \)
    - \( BF_{3,i}(1, v) = BF_{i,3}(u, 1) = 0 \)
    - \( BF_{1,i}(1, v) = BF_{i,1}(u, 1) = 0 \)
    - \( BF_{2,i}(1, v) = BF_{i,2}(u, 1) = 0 \)
    - \( BF_{3,i}(1, v) = BF_{i,3}(u, 1) = 0 \)

    If \( BF_0(1) = 0 \), then since:
      \[ BF_i(u, v) - BF_i(u, 1) - BF_i(1, v) = 0 \]
    In a similar fashion you can show that if the curve blending function \( BF_i(u) \) satisfy the continuity conditions, then so do the \( BF_{i,j}(u, v) \).

    More generally, if the \( BF_i(u) \) give continuous \( n \)-th order derivatives, then so will the \( BF_{i,j}(u, v) \).
Surface Spline Properties

Convex hull containment:

• We need the weights of the blending function to be non-negative.

• If the $BF_i(u)$ are non-negative, then since

$$BF_{i,j}(u, v) = BF_i(u) \cdot BF_j(v)$$

the $BF_{i,j}(u, v)$ will also be non-negative.

• So the patch spline will also be in the convex hull.
Surface Spline Properties

Interpolation:

• If the spline curve is interpolating, then it satisfies:
  
  \[ BF_0(0) = BF_2(0) = BF_3(0) = 0 \]
  \[ BF_0(1) = BF_1(1) = BF_3(1) = 0 \]
  \[ BF_1(0) = 1 \]
  \[ BF_2(1) = 1 \]

But then at the end-points (0,0), (1,0), (0,1) and (1,1) we must have:

  \[ BF_{1,1}(0,0) = BF_{1,2}(0,1) = BF_{2,1}(1,0) = BF_{2,2}(1,1) = 1. \]
  \[ \text{All the other blending functions are 0 at the end-points.} \]

• So the patch spline will also interpolate.
Surface Spline Properties

We began by describing some of the properties that we would like spline curves to satisfy:

- Translation commutativity
- Continuity
- Convex hull containment
- Interpolation

If the curve spline satisfies these properties, then so will the tensor product spline!

Spline Demo