Parametric Curves

Thomas Funkhouser
Princeton University
COS 426, Spring 2006

3D Object Representations

- Raw data
  - Voxels
  - Point cloud
  - Range image
  - Polygons

- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit

- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep

- High-level structures
  - Scene graph
  - Application specific

Parametric Surfaces

- Boundary defined by parametric functions:
  - \( x = f_1(u,v) \)
  - \( y = f_2(u,v) \)
  - \( z = f_3(u,v) \)

- Example: ellipsoid
  - \( x = r \cos \phi \cos \theta \)
  - \( y = r \cos \phi \sin \theta \)
  - \( z = r \sin \phi \)

- Parametric functions define mapping from \((u,v)\) to \((x,y,z)\)

H&B Figure 10.10

H&B Figure 10.46

Princeton University

Parametric Surfaces

- Boundary defined by parametric functions:
  - \( x = f_1(u,v) \)
  - \( y = f_2(u,v) \)
  - \( z = f_3(u,v) \)

- Design of smooth surfaces in cars, ships, etc.

H&B Figure 10.46

F&DH Figure 11.42
**Parametric Curves**

- Boundary defined by parametric functions:
  - \( x = f_1(u) \)
  - \( y = f_2(u) \)

- Example: ellipse
  - \( x = r \cos \theta \)
  - \( y = r \sin \theta \)

**Implicit curves**

An implicit curve in the plane is expressed as:

\[ f(x, y) = 0 \]

**Example:** a circle with radius \( r \) centered at origin:

\[ x^2 + y^2 - r^2 = 0 \]

**Curves in Computer Graphics**

- Fonts **ABC**
- Animation paths
- Shape modeling
- etc...

**Parametric curves**

How can we define arbitrary curves?

\[
\begin{align*}
x &= f_1(u) \\
y &= f_2(u)
\end{align*}
\]

Use functions that "blend" control points

\[
\begin{align*}
x &= f_1(u) = V_0^\ast(1 - u) + V_1^\ast u \\
y &= f_2(u) = V_0^\ast(1 - u) + V_1^\ast u
\end{align*}
\]

**Parametric curves**

More generally:

\[
\begin{align*}
x(u) &= \sum_{i=0}^{n} B_i(u) \ast V_i \\
y(u) &= \sum_{i=0}^{n} B_i(u) \ast V_i
\end{align*}
\]
Continuity

- Parametric continuity ($C^n$)
  - How many times differentiable is the curve with respect to $u$ at a given point
- Parametric continuity at joints:
  - $C^0$ continuity means curve is connected at joint
  - $C^1$ continuity means that segments share same first derivative at joint
  - $C^n$ continuity means that segments share same $n$th derivative at joint
- Relationships:
  - $C^n$ implies $C^{n-1}$

Goals

- Some attributes we might like to have:
  - Efficient computation
  - Predictable control
  - Local control
  - Interpolation
  - Continuity

Parametric curves

What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i$$

$$y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i$$

Continuity

- Geometric continuity ($G^n$)
  - How many times differentiable is the curve with respect to $x,y$ at a given point
- Relationships:
  - $C^n$ implies $G^n$, but not vice-versa
Cubic Piecewise Parametric Polynomial Curves

- Blending functions are polynomials:
  \[
  x(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i \\
  y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i
  \]
  
- Advantages of polynomials
  - Easy to compute
  - Infinitely continuous
  - Easy to derive curve properties

Goals

- Some attributes we might like to have:
  - Efficient computation
  - Predictable control
  - Local control
  - Interpolation
  - Continuity

- We’ll satisfy these goals using:
  - Piecewise
  - Parametric
  - Polynomials

Parametric Polynomial Curves

- Blending functions are polynomials:
  \[
  Q(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i \\
  B_i(u) = \sum_{j=0}^{m} a_j \cdot u^j
  \]

- Advantages of polynomials
  - Easy to compute
  - Infinitely continuous
  - Easy to derive curve properties

Parametric Polynomial Curves

- Splines:
  - Split curve into segments
  - Each segment defined by low-order polynomial blending subset of control vertices

- Motivation:
  - Provides control & efficiency
  - Same blending function for every segment
  - Prove properties from blending functions

- Challenges
  - How choose blending functions?
  - How guarantee continuity at joints?

Piecewise Parametric Polynomial Curve

- Compute polynomial \( B_i(u) \) to ensure properties
  - Example: Interpolation of control vertices and \( C^2 \) continuity at joints with cubics

Cubic Piecewise Parametric Polynomial Curve

- From now on, consider cubic blending functions
  - All ideas generalize to higher degrees

- In CAGD, higher-order functions are often used
  - Hard to control wiggles

- In graphics, piecewise cubic curves will do
  - Smallest degree that allows \( C^2 \) continuity for arbitrary curves
Natural Cubic Hermite Splines

- Definition: 4\(n\)-1 degrees of freedom
- Properties:
  - \(C^1\) continuity
  - 2\((n-1)\) constraints: \(Q'(1)=Q_{n+1}'(0)\) and \(Q''(1)=Q_{n+1}''(0)\)
- Solve system of equation for coefficients of blending functions

Cubic Hermite Splines

- Definition: 4\(n\)-1 degrees of freedom
- Properties:
  - \(C^1\) continuity
  - 2\((n-2)\) constraints: \(Q_i(0)=V_i\) and \(Q_{i+1}(1)=V_{i+1}\)

Cubic Hermite Splines

- Definition:
  - Each segment defined by position and derivative at two adjacent control vertices
  - Blending functions are cubic polynomials
  - \(4(n-1)\) degrees of freedom

Each segment defined by cubic Hermite Splines

Each has different blending functions resulting in different properties

Types of Splines

- Splines covered in this lecture
  - Hermite
  - B-Spline
  - Bezier
- There are many others
• Problems:
  - No local control
  - Whole curve adjusts to any movement of control vertex
  - Every segment has different blending functions
  - Hard to prove properties

**Uniform Cubic Hermite Splines**

- Properties:
  - Interpolates control points
  - Same blending function for every segment
  - Local control
  - C^1 continuity at joints

**Types of Spline Curves**

- Splines covered in this lecture
  - Hermite
  - B-Spline
  - Bezier

- There are many others

Each has different blending functions resulting in different properties

**Uniform Cubic B-Splines**

- Properties:
  - Local control
  - C^0 continuity
  - Approximating

**B-Spline Blending Functions**

- Properties imply blending functions:
  - Cubic polynomials
  - Four control vertices affect each point
  - C^2 continuity
B-Spline Blending Functions

• How derive blending functions?
  - Cubic polynomials
  - Local control
  - C^2 continuity

B-Spline Blending Functions

• Four cubic polynomials for four vertices
  - 16 variables (degrees of freedom)
  - Variables are \( a, b, c, d \) for four blending functions

\[
\begin{align*}
    b_0(u) &= a_0 u^3 + b_0 u^2 + c_0 u + d_0 \\
    b_1(u) &= a_1 u^3 + b_1 u^2 + c_1 u + d_1 \\
    b_2(u) &= a_2 u^3 + b_2 u^2 + c_2 u + d_2 \\
    b_3(u) &= a_3 u^3 + b_3 u^2 + c_3 u + d_3
\end{align*}
\]

B-Spline Blending Functions

• C^2 continuity implies 15 constraints
  - Position of two curves same
  - Derivative of two curves same
  - Second derivatives same

B-Spline Blending Functions

Fifteen continuity constraints:

\[
\begin{align*}
    0 &= b_0(0) \\
    0 &= b_0'(0) \\
    0 &= b_0''(0) \\
    b_0(1) &= b_1(0) \\
    b_0'(1) &= b_1'(0) \\
    b_0''(1) &= b_1''(0) \\
    b_0(1) - b_1(0) &= b_1'(0) - b_1'(0) \\
    b_0''(1) - b_1''(0) &= b_1''(0) - b_1''(0) \\
    b_0(1) &= b_2(0) \\
    b_0'(1) &= b_2'(0) \\
    b_0''(1) &= b_2''(0) \\
    b_0(1) - b_2(0) &= b_2'(0) - b_2'(0) \\
    b_0''(1) - b_2''(0) &= b_2''(0) - b_2''(0) \\
    b_0(1) &= b_3(0) \\
    b_0'(1) &= b_3'(0) \\
    b_0''(1) &= b_3''(0) \\
    b_0(1) - b_3(0) &= b_3'(0) - b_3'(0) \\
    b_0''(1) - b_3''(0) &= b_3''(0) - b_3''(0)
\end{align*}
\]

One more convenient constraint:

\[
    b_0(0) + b_1(0) + b_2(0) + b_3(0) - 1
\]

B-Spline Blending Functions

• Solving the system of equations yields:

\[
\begin{align*}
    b_0(u) &= \frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6} \\
    b_1(u) &= \frac{1}{2}u^3 - u^2 + \frac{1}{3} \\
    b_2(u) &= \frac{1}{2}u^3 - u^2 + \frac{1}{2}u + \frac{1}{6} \\
    b_3(u) &= \frac{1}{6}u^3
\end{align*}
\]

B-Spline Blending Functions

• In matrix form:

\[
Q(u) = \begin{bmatrix} u^3 & u^2 & u \end{bmatrix} ^T \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix}
\]
Basic properties of Bézier curves

- Endpoint interpolation:
  \[ Q(0) = V_0 \]
  \[ Q(1) = V_n \]

- Convex hull:
  - Curve is contained within convex hull of control polygon

- Symmetry
  - \( Q(\alpha) \) defined by \( \{V_0, \ldots, V_n\} \) = \( Q(1-\alpha) \) defined by \( \{V_n, \ldots, V_0\} \)

Matrix form

Bézier curves may be described in matrix form:

\[
Q(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} V_i = \begin{pmatrix} 1 & -3 & 3 & -1 \end{pmatrix} M_{\text{Bezier}} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}
\]

Types of Splines

- Spline covered in this lecture
  - Hermite
  - B-Spline
  - Bezier

- There are many others

B-Spline Blending Functions

- Blending functions imply properties:
  - Local control
  - Approximating
  - \( C^2 \) continuity
  - Convex hull

B-Spline Blending Functionsss

In plot form:

\[
B_i(u) = \sum_{j=0}^{n} a_{ij} u^j
\]

Each has different blending functions resulting in different properties
**Bézier curves**

- Curve \( Q(u) \) can also be defined by nested interpolation:

\[
V_0, V_1, ..., V_n \text{ are control points}
\]

\( \{V_0, V_1, ..., V_n\} \text{ is control polygon} \)

**Display**

Q: How would you draw it using line segments?
A: Recursive subdivision!

**Display**

Pseudocode for displaying Bézier curves:

```plaintext
procedure Display(V_i):
  if \( \{V_i\} \) flat within \( \varepsilon \)
    then output line segment \( V_0V_i \)
    else subdivide to produce \( \{L_i\} \) and \( \{R_i\} \)
    Display(\( \{L_i\} \))
    Display(\( \{R_i\} \))
  end if
end procedure
```

**Flatness**

Q: How do you test for flatness?
A: Compare the length of the control polygon to the length of the segment between endpoints:

\[
\frac{|V_i - V_0| + |V_i - V_1| + |V_i - V_n|}{|V_i - V_0|} < 1 + \varepsilon
\]

**Beziers Splines**

- For more complex curves, piece together Bézier curves

- Solve for “interior” control vertices
  - Positional \( (C^0) \) continuity
  - Derivative \( (C^1) \) continuity
Summary

• Splines: mathematical way to express curves
• Motivated by “loftsman’s spline”
  ▶ Long, narrow strip of wood/plastic
  ▶ Used to fit curves through specified data points
  ▶ Shaped by lead weights called “ducks”
  ▶ Gives curves that are “smooth” or “fair”
• Have been used to design:
  ▶ Automobiles
  ▶ Ship hulls
  ▶ Aircraft fuselage/wing

What’s next?

• Use curves to create parameterized surfaces