



Parametric Curves

Thomas Funkhouser
Princeton University
COS 426, Spring 2006



3D Object Representations

- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- Surfaces
 - Mesh
 - Subdivision
 - Parametric
 - Implicit
- High-level structures
 - Scene graph
 - Application specific



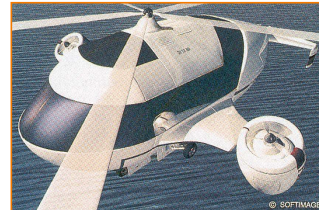
3D Object Representations

- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- Surfaces
 - Mesh
 - Subdivision
 - **Parametric**
 - Implicit
- High-level structures
 - Scene graph
 - Application specific



Parametric Surfaces

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



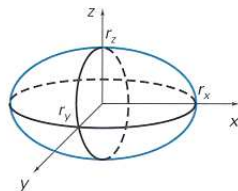
H&B Figure 10.46



Parametric Surfaces

- Boundary defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

- Example: ellipsoid
 - $x = r_x \cos \phi \cos \theta$
 - $y = r_y \cos \phi \sin \theta$
 - $z = r_z \sin \phi$

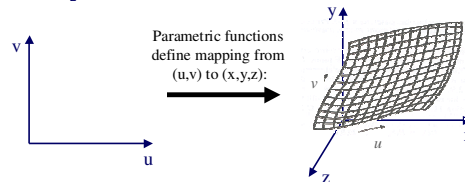


H&B Figure 10.10



Parametric Surfaces

- Boundary defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$



FvDPH Figure 11.42

Parametric Curves



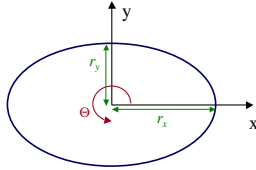
- Boundary defined by parametric functions:

- $x = f_x(u)$
- $y = f_y(u)$

- Example: ellipse

$$x = r_x \cos \theta$$

$$y = r_y \sin \theta$$



H&B Figure 10.10

Implicit curves

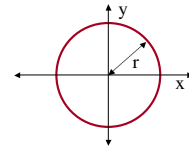


An implicit curve in the plane is expressed as:

$$f(x, y) = 0$$

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?

$$x = f_x(u)$$

$$y = f_y(u)$$



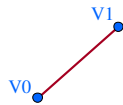
Parametric curves



How can we define arbitrary curves?

$$x = f_x(u)$$

$$y = f_y(u)$$



Use functions that "blend" control points

$$x = f_x(u) = V0_x * (1 - u) + V1_x * u$$

$$y = f_y(u) = V0_y * (1 - u) + V1_y * u$$

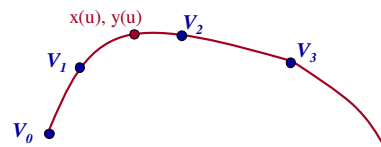
Parametric curves



More generally:

$$x(u) = \sum_{i=0}^n B_i(u) * V_{i_x}$$

$$y(u) = \sum_{i=0}^n B_i(u) * V_{i_y}$$



Parametric curves



What B(u) functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$

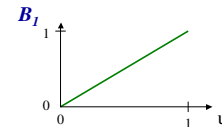
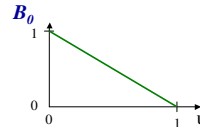
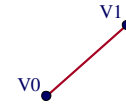
Parametric curves



What B(u) functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$



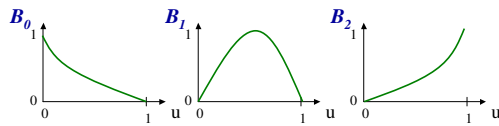
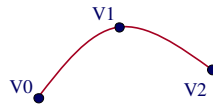
Parametric curves



What B(u) functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$



Goals



Some attributes we might like to have:

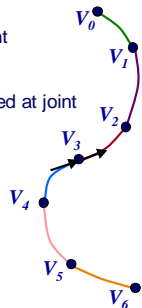
- Efficient computation
- Predictable control
- Local control
- Interpolation
- Continuity



Continuity



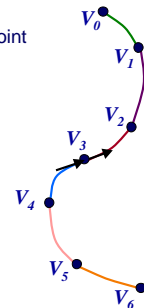
- Parametric continuity (C^n)
 - How many times differentiable is the curve with respect to u at a given point
- Parametric continuity at joints:
 - C^0 continuity means curve is connected at joint
 - C^1 continuity means that segments share same first derivative at joint
 - C^n continuity means that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



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



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:

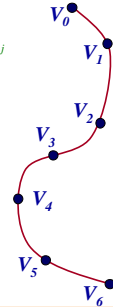
$$x(u) = \sum_{i=0}^n B_i(u) * V_{i_x}$$

$$y(u) = \sum_{i=0}^n B_i(u) * V_{i_y}$$

$$B_i(u) = \sum_{j=0}^m a_j u^j$$

- Advantages of polynomials

- Easy to compute
- Infinitely continuous
- Easy to derive curve properties



Parametric Polynomial Curves

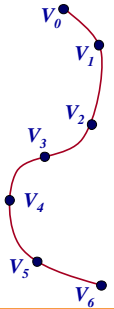
- Blending functions are polynomials:

$$Q(u) = \sum_{i=0}^n B_i(u) * V_i$$

$$B_i(u) = \sum_{j=0}^m a_j u^j$$

- Advantages of polynomials

- Easy to compute
- Infinitely continuous
- Easy to derive curve properties



Piecewise Parametric Polynomial Curves

- Splines:

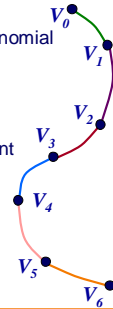
- Split curve into segments
- Each segment defined by low-order polynomial blending subset of control vertices

- Motivation:

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

- Challenges

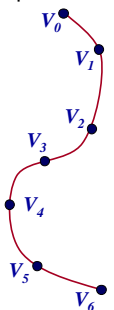
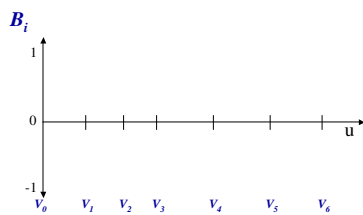
- How choose blending functions?
- How guarantee continuity at joints?



Piecewise Parametric Polynomial Curves

- 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 Curves

- 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

Types of Splines



- Splines covered in this lecture
 - Hermite
 - B-Spline
 - Bezier
- There are many others

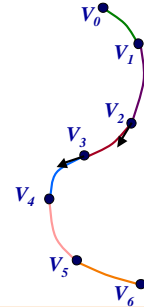
Each has different blending functions resulting in different properties

Cubic Hermite Splines



- Definition:
 - Each segment defined by position and derivative at two adjacent control vertices
 - Blending functions are cubic polynomials

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$
 - $4(n-1)$ degrees of freedom



Cubic Hermite Splines



- Definition:
 - Each segment defined by position and derivative at two adjacent control vertices
 - Blending functions are cubic polynomials

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$



$$Q(u) = B_0(u)*V_0 + B_1(u)*V_1 + B_2(u)*D_0 + B_3(u)*D_1$$

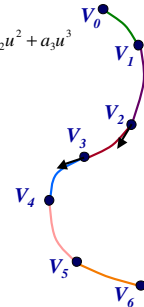
Cubic Hermite Splines



- Definition:
 - $4(n-1)$ degrees of freedom
- Properties:
 - Interpolates control points
 - $2(n-2)$ constraints:

$$Q_i(0) = V_i \text{ and } Q_i(1) = V_{i+1}$$

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$



Natural Cubic Hermite Splines

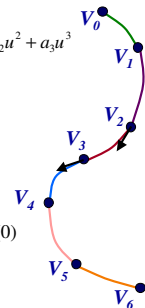


- Definition:
 - $4(n-1)$ degrees of freedom
- Properties:
 - Interpolates control points
 - $2(n-2)$ constraints:

$$Q_i(0) = V_i \text{ and } Q_i(1) = V_{i+1}$$
 - C^2 continuity
 - $2(n-1)$ constraints:

$$Q_i'(1) = Q_{i+1}'(0) \text{ and } Q_i''(1) = Q_{i+1}''(0)$$

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$



Natural Cubic Hermite Splines

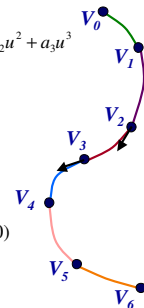


- Definition:
 - $4(n-1)$ degrees of freedom
- Properties:
 - Interpolates control points
 - $2(n-2)$ constraints:

$$Q_i(0) = V_i \text{ and } Q_i(1) = V_{i+1}$$
 - C^2 continuity
 - $2(n-1)$ constraints:

$$Q_i'(1) = Q_{i+1}'(0) \text{ and } Q_i''(1) = Q_{i+1}''(0)$$

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$

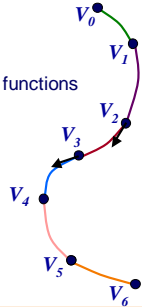


Solve system of equation for coefficients of blending functions

Natural Cubic Hermite Splines



- Problems:
 - No local control
 - Whole curve adjusts to any movement of control vertex
 - Every segment has different blending functions
 - Hard to prove properties



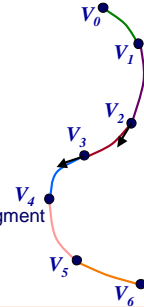
Uniform Cubic Hermite Splines



- Definition:
 - Each segment defined by position and derivative at two adjacent control vertices
 - Blending functions are cubic polynomials

$$B(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$

- Properties:
 - Interpolates control points
 - Same blending function for every segment
 - Local control
 - C¹ continuity at joints

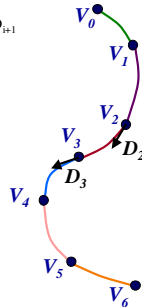
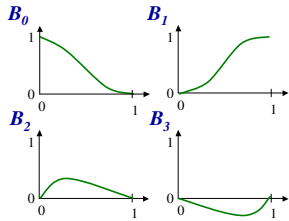


Uniform Cubic Hermite Splines



Blending functions:

$$Q(u) = B_0(u)*V_0 + B_1(u)*V_1 + B_2(u)*D_1 + B_3(u)*D_2$$



Types of Spline Curves



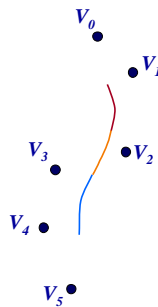
- Splines covered in this lecture
 - Hermite
 - B-Spline
 - Bezier
- There are many others

Each has different blending functions resulting in different properties

Uniform Cubic B-Splines



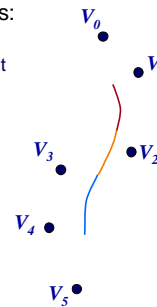
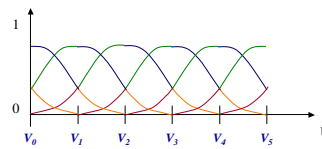
- Properties:
 - Local control
 - C² continuity
 - Approximating



B-Spline Blending Functions



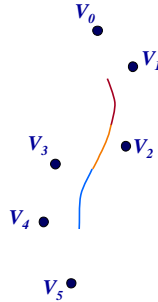
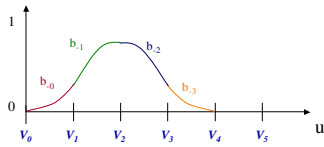
- Properties imply blending functions:
 - Cubic polynomials
 - Four control vertices affect each point
 - C² continuity



B-Spline Blending Functions



- How derive blending functions?
 - Cubic polynomials
 - Local control
 - C² continuity

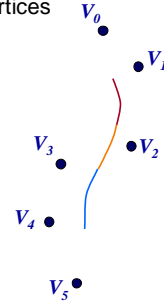


B-Spline Blending Functions



- Four cubic polynomials for four vertices
 - 16 variables (degrees of freedom)
 - Variables are a_i, b_i, c_i, d_i for four blending functions

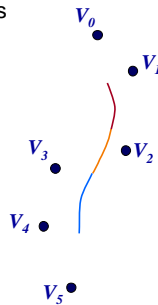
$$\begin{aligned} b_{-0}(u) &= a_0u^3 + b_0u^2 + c_0u + d_0 \\ b_{-1}(u) &= a_1u^3 + b_1u^2 + c_1u + d_1 \\ b_{-2}(u) &= a_2u^3 + b_2u^2 + c_2u + d_2 \\ b_{-3}(u) &= a_3u^3 + b_3u^2 + c_3u + d_3 \end{aligned}$$



B-Spline Blending Functions



- C2 continuity implies 15 constraints
 - Position of two curves same
 - Derivative of two curves same
 - Second derivatives same



B-Spline Blending Functions



Fifteen continuity constraints:

$$\begin{aligned} 0 &= b_{-0}(0) & 0 &= b_{-0}'(0) & 0 &= b_{-0}''(0) \\ b_{-0}(1) &= b_{-1}(0) & b_{-0}'(1) &= b_{-1}'(0) & b_{-0}''(1) &= b_{-1}''(0) \\ b_{-1}(1) &= b_{-2}(0) & b_{-1}'(1) &= b_{-2}'(0) & b_{-1}''(1) &= b_{-2}''(0) \\ b_{-2}(1) &= b_{-3}(0) & b_{-2}'(1) &= b_{-3}'(0) & b_{-2}''(1) &= b_{-3}''(0) \\ b_{-3}(1) &= 0 & b_{-3}'(1) &= 0 & b_{-3}''(1) &= 0 \end{aligned}$$

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:

$$\begin{aligned} b_{-3}(u) &= -\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6} \\ b_{-2}(u) &= \frac{1}{2}u^3 - u^2 + \frac{2}{3} \\ b_{-1}(u) &= -\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6} \\ b_{-0}(u) &= \frac{1}{6}u^3 \end{aligned}$$

B-Spline Blending Functions



- In matrix form:

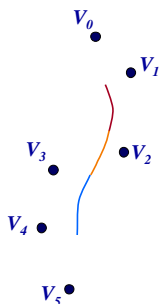
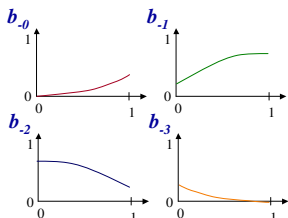
$$Q(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

B-Spline Blending Functions



In plot form:

$$B_i(u) = \sum_{j=0}^m a_j u^j$$

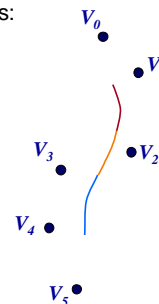
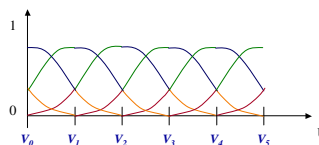


B-Spline Blending Functions



Blending functions imply properties:

- Local control
- Approximating
- C² continuity
- Convex hull



Types of Splines



Splines covered in this lecture

- Hermite
- B-Spline
- ∅ Bezier

There are many others

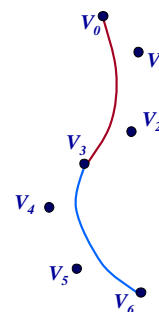
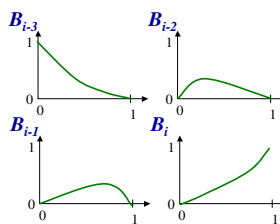
Each has different blending functions resulting in different properties

Bezier curves



Blending functions:

$$B_i(u) = \sum_{j=0}^m a_j u^j$$



Matrix form



Bézier curves may be described in matrix form:

$$\begin{aligned} Q(u) &= \sum_{i=0}^n V_i \binom{n}{i} u^i (1-u)^{n-i} \\ &= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\ &= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \end{aligned}$$

M_{Bezier}

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

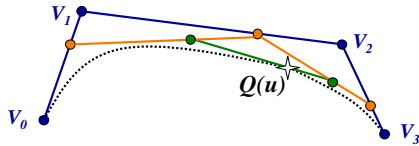
Symmetry

$$Q(u) \text{ defined by } \{V_0, \dots, V_n\} \equiv Q(1-u) \text{ defined by } \{V_n, \dots, V_0\}$$

Bézier curves



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



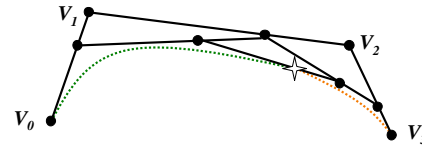
V_i 's are control points
 $\{V_0, V_1, \dots, V_n\}$ is control polygon

Display



Q: How would you draw it using line segments?

A: Recursive subdivision!



Display



Pseudocode for displaying Bézier curves:

```

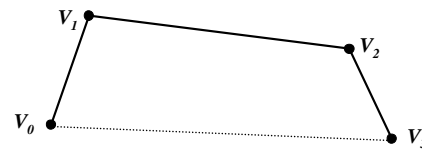
procedure Display( $\{V_i\}$ ):
  if  $\{V_i\}$  flat within  $\epsilon$ 
  then
    output line segment  $V_0V_n$ 
  else
    subdivide to produce  $\{L_i\}$  and  $\{R_i\}$ 
    Display( $\{L_i\}$ )
    Display( $\{R_i\}$ )
  end if
end procedure
  
```

Flatness



Q: How do you test for flatness?

A: Compare the length of the control polygon to the length of the segment between endpoints

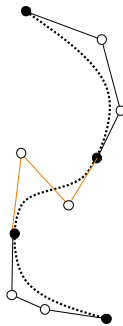


$$\frac{|V_1 - V_0| + |V_2 - V_1| + |V_3 - V_2|}{|V_3 - V_0|} < 1 + \epsilon$$

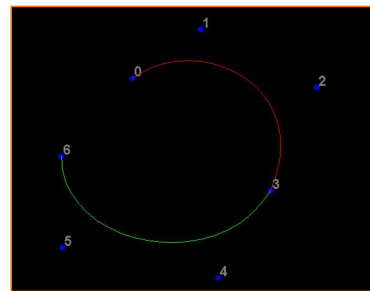
Bezier Splines



- For more complex curves, piece together Bézier curves
- Solve for "interior" control vertices
 - Positional (C^0) continuity
 - Derivative (C^1) continuity



Bezier Splines



Patrick Min

Summary

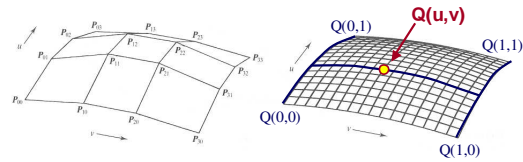


- Splines: mathematical way to express curves
- Motivated by “loftman’s spline”
 - Long, narrow strip of wood/plastic
 - Used to fit curves through specified data points
 - Shaped by lead weights called “ducks”
 - Gives curves that are “smooth” or “fair”
- Have been used to design:
 - Automobiles
 -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

What’s next?



- Use curves to create parameterized surfaces



Watt Figure 6.21