Advanced Computer Graphics
CSCI B581 – Spring 2017

Lecture 12 – 2017-02-20

Instructor:
Mitja Hmeljak,
http://pages.iu.edu/~mitja
mitja@indiana.edu
Splines, Curves and Surfaces

a. Curves

b. Designing Parametric Cubic Curves

c. Bezier and B-Spline Curves
a. Curves and Surfaces

Introduce types of curves
- Explicit
- Implicit
- Parametric
- Strengths and weaknesses

Discuss Modeling and Approximations
- Conditions
- Stability
Escaping Flatland

Working with *flat* entities such as lines:
- Fits well with graphics hardware
- Mathematically simple

But the world is not composed of flat entities
- Need curves (and curved surfaces)
- May only have need at the application level
- Implementation can render them approximately with flat primitives
Modeling with Curves

- Data points
- Approximating curve
- Interpolating data point
What Makes a Good Representation?

There are many ways to represent curves and surfaces

Want a representation that is
- Stable
- Smooth
- Easy to evaluate

Must we interpolate or can we just come close to data?

Do we need derivatives?
Explicit Representation

Most familiar form of curve in 2D

\[ y = f(x) \]

Cannot represent all curves, e.g.:
- Vertical lines
- Circles

(Extension to 3D:
\[ y = f(x), \]
\[ z = g(x) \]
\[ \Rightarrow \text{the form } z = f(x,y) \text{ defines a surface} \]
Implicit Representation

Two dimensional curve(s), implicit representation:

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

... much more robust than explicit representation.

All lines \( ax + by + c = 0 \)

Circles \( x^2 + y^2 - r^2 = 0 \)

In 3D, \( g(x,y,z) = 0 \) defines a surface

We can intersect two surfaces to get a curve

In general, we can not solve for points that satisfy.
Algebraic Surface

\[ \sum_i \sum_j \sum_k x^i y^j z^k = 0 \]

- Quadric surface \( 2 \geq i+j+k \)
  - (points where a quadratic polynomial becomes \( == 0 \))
- At most 10 terms

Can solve *intersection with a ray* by reducing this problem to *solving a quadratic equation*.
Parametric Curves

Separate equation for each spatial variable

\[ x = x(u) \]
\[ y = y(u) \]
\[ z = z(u) \]

For \( u_{\text{max}} \geq u \geq u_{\text{min}} \) we trace out a curve in two or three dimensions

\[ \mathbf{p}(u) = [x(u), y(u), z(u)]^T \]
Selecting Functions

Usually we can select “good” functions not unique for a given spatial curve. Approximate or interpolate known data. Want functions which are easy to evaluate. Want functions which are easy to differentiate. Computation of normals. Connecting pieces (segments). Want functions which are smooth.
We can normalize $u$ to be over the interval $(0,1)$

Line connecting two points $\mathbf{p}_0$ and $\mathbf{p}_1$

$$p(u) = (1-u) \mathbf{p}_0 + u \mathbf{p}_1$$

Ray from $\mathbf{p}_0$ in the direction $\mathbf{d}$

$$p(u) = \mathbf{p}_0 + u \mathbf{d}$$
Parametric Surfaces

Surfaces require 2 parameters

\[ x = x(u,v) \]
\[ y = y(u,v) \]
\[ z = z(u,v) \]

\[ p(u,v) = [x(u,v), y(u,v), z(u,v)]^T \]

Want same properties as curves:

Smoothness
Differentiability
Ease of evaluation
Normals

We can differentiate with respect to $u$ and $v$ to obtain the normal at any point $p$

\[
\frac{\partial p(u, v)}{\partial u} = \begin{bmatrix}
\frac{\partial x(u, v)}{\partial u} \\
\frac{\partial y(u, v)}{\partial u} \\
\frac{\partial z(u, v)}{\partial u}
\end{bmatrix}
\]

\[
\frac{\partial p(u, v)}{\partial v} = \begin{bmatrix}
\frac{\partial x(u, v)}{\partial v} \\
\frac{\partial y(u, v)}{\partial v} \\
\frac{\partial z(u, v)}{\partial v}
\end{bmatrix}
\]

\[
\mathbf{n} = \frac{\partial p(u, v)}{\partial u} \times \frac{\partial p(u, v)}{\partial v}
\]
Parametric Planes

point-vector form
\[ p(u,v) = p_0 + uq + vr \]
\[ n = q \times r \]

three-point form
\[ q = p_1 - p_0 \]
\[ r = p_2 - p_0 \]
How do we define a Sphere in Parametric Form?

spherical coordinate system:

\[ x(u,v) = r \cos \theta \sin \phi \]
\[ y(u,v) = r \sin \theta \sin \phi \]
\[ z(u,v) = r \cos \phi \]

\[ 360 \geq \theta \geq 0 \]
\[ 180 \geq \phi \geq 0 \]

\( \theta \) constant: circles of constant longitude
\( \phi \) constant: circles of constant latitude

differentiate to show \( \mathbf{n} = \mathbf{p} \)
Curve Segments as Parametric Curves:

After normalizing \( u \), each curve is written

\[
p(u) = [x(u), y(u), z(u)]^T, \quad 1 \geq u \geq 0
\]

In classical numerical methods, we design a single global curve.

In computer graphics and CAD, it is better to design small connected curve segments.

\[ p(0) \rightarrow p(1) \quad \text{join point } p(1) = q(0) \quad q(0) \rightarrow q(1) \]
Parametric Polynomial Curves

\[ x(u) = \sum_{i=0}^{N} c_{xi} u^i \quad y(u) = \sum_{j=0}^{M} c_{yj} u^j \quad z(u) = \sum_{k=0}^{L} c_{zk} u^k \]

• If N=M=K, we need to determine 3(N+1) coefficients

• Equivalently we need 3(N+1) independent conditions

• Noting that the curves for x, y and z are independent, we can define each independently in an identical manner

• We will use the form where p can be any of x, y, z

\[ p(u) = \sum_{k=0}^{L} c_{k} u^{k} \]
Why Polynomials

Easy to evaluate

Continuous and differentiable everywhere

Must worry about continuity at join points including continuity of derivatives

$p(u)$

$q(u)$

join point $p(1) = q(0)$ but $p'(1) \neq q'(0)$
Cubic Parametric Polynomials: order 3

N=M=L=3, gives balance between ease of evaluation and flexibility in design

\[ p(u) = \sum_{k=0}^{3} c_k u^k \]

Four coefficients to determine for each of x, y and z

Seek four independent conditions for various values of u resulting in 4 equations in 4 unknowns for each of x, y and z.

Conditions are a mixture of continuity requirements at the join points and conditions for fitting the data.
Cubic Polynomial Surfaces

\[ \mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T \]

where

\[ \mathbf{p}(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} c_{ij} u^i v^j \]

\( \mathbf{p} \) is any of \( x, y \) or \( z \)

Need 48 coefficients (3 independent sets of 16) to determine a surface patch
b. Designing Parametric Cubic Curves

Types of curves:
- Interpolating
- Hermite
- Bezier
- B-spline

Analyze and compare these types of curves
Matrix-Vector Form

\[ p(u) = \sum_{k=0}^{3} c_k u^k \]

define \( c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \) \( u = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \)

then \( p(u) = u^T c = c^T u \)
Defining an Interpolating Curve

Given four data (control) points $p_0, p_1, p_2, p_3$ determine cubic $p(u)$ which passes through them.

Must find $c_0, c_1, c_2, c_3$
Interpolation Equations

apply the interpolating conditions at \( u=0, 1/3, 2/3, 1 \)

\[
p_0 = p(0) = c_0 \\
p_1 = p(1/3) = c_0 + (1/3)c_1 + (1/3)^2c_2 + (1/3)^3c_2 \\
p_2 = p(2/3) = c_0 + (2/3)c_1 + (2/3)^2c_2 + (2/3)^3c_2 \\
p_3 = p(1) = c_0 + c_1 + c_2 + c_2
\]

or in matrix form with \( \mathbf{p} = [p_0 \ p_1 \ p_2 \ p_3]^T \)

\[\mathbf{p} = \mathbf{A}\mathbf{c}\]

\[
\mathbf{A} = \\
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & (1/3) & (1/3)^2 & (1/3)^3 \\
1 & (2/3) & (2/3)^2 & (2/3)^3 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]
Interpolation Matrix

Solving for \( \mathbf{c} \) (we need the values of all \( c \) coefficients) we find the interpolation matrix

\[
\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-5.5 & 9 & -4.5 & 1 \\
9 & -22.5 & 18 & -4.5 \\
-4.5 & 13.5 & -13.5 & 4.5
\end{bmatrix}
\]

\( \mathbf{c} = \mathbf{M}_I \mathbf{p} \)

Note that \( \mathbf{M}_I \) does not depend on input data and can be used for each segment in \( x, y, \) and \( z \)
Interpolating Multiple Segments

use \( \mathbf{p} = [p_0 \ p_1 \ p_2 \ p_3]^T \)

use \( \mathbf{p} = [p_3 \ p_4 \ p_5 \ p_6]^T \)

Get continuity at join points but not continuity of derivatives
Blending Functions

Rewriting the equation for \( p(u) \)

\[
p(u) = u^T c = u^T M_I p = b(u)^T p
\]

where \( b(u) = [b_0(u) \ b_1(u) \ b_2(u) \ b_3(u)]^T \) is an array of \textit{blending polynomials} such that

\[
p(u) = b_0(u)p_0 + b_1(u)p_1 + b_2(u)p_2 + b_3(u)p_3
\]

\[
b_0(u) = -4.5(u-1/3)(u-2/3)(u-1)
b_1(u) = 13.5u \ (u-2/3)(u-1)
b_2(u) = -13.5u \ (u-1/3)(u-1)
b_3(u) = 4.5u \ (u-1/3)(u-2/3)
\]
Blending Functions

These functions are not smooth
Hence the interpolation polynomial is not smooth
Interpolating Patch

\[ p(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} c_{ij} u^i v^j \]

Need 16 conditions to determine the 16 coefficients \( c_{ij} \)
Choose at \( u,v = 0, 1/3, 2/3, 1 \)
Matrix Form

Define \( \mathbf{v} = [1 \ v \ v^2 \ v^3]^T \)

\[
\mathbf{C} = [c_{ij}] \quad \mathbf{P} = [p_{ij}]
\]

\( p(u,v) = \mathbf{u}^T \mathbf{C} \mathbf{v} \)

If we observe that for constant \( u \) (\( v \)), we obtain interpolating curve in \( v \) (\( u \)), we can show

\[
\mathbf{C} = \mathbf{M}_I \mathbf{P} \mathbf{M}_I^T
\]

\( p(u,v) = \mathbf{u}^T \mathbf{M}_I \mathbf{P} \mathbf{M}_I^T \mathbf{v} \)
Blending Patches

\[ p(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u)b_j(v)p_{ij} \]

Each \( b_i(u)b_j(v) \) is a blending patch

Shows that we can build and analyze surfaces from our knowledge of curves
Other Types of Curves and Surfaces

How can we get around the limitations of the interpolating form
Lack of smoothness
Discontinuous derivatives at join points
We have four conditions (for cubics) that we can apply to each segment
Use them other than for interpolation
Need only come close to the data
Hermite Form

Use two interpolating conditions and two derivative conditions per segment.

Ensures continuity and first derivative continuity between segments.
Interpolating conditions are the same at ends

\[ p(0) = p_0 = c_0 \]
\[ p(1) = p_3 = c_0 + c_1 + c_2 + c_3 \]

Differentiating we find \( p'(u) = c_1 + 2uc_2 + 3u^2c_3 \)

Evaluating at end points

\[ p'(0) = p'_0 = c_1 \]
\[ p'(1) = p'_3 = c_1 + 2c_2 + 3c_3 \]
Solving, we find $\mathbf{c} = \mathbf{M}_H \mathbf{q}$ where $\mathbf{M}_H$ is the Hermite matrix.
Blending Polynomials

\[ p(u) = b(u)^T q \]

\[ b(u) = \begin{bmatrix}
2u^3 - 3u^2 + 1 \\
-2u^3 + 3u^2 \\
u^3 - 2u^2 + u \\
u^3 - u^2
\end{bmatrix} \]

Although these functions are smooth, the Hermite form is not used directly in Computer Graphics and CAD because we usually have control points but not derivatives.

However, the Hermite form is the basis of the Bezier form.
Parametric and Geometric Continuity

We can require the derivatives of \( x, y, \) and \( z \) to each be continuous at join points (parametric continuity).

Alternately, we can only require that the tangents of the resulting curve be continuous (geometry continuity).

The latter gives more flexibility as we have need satisfy only two conditions rather than three at each join point.
Example

Here the p and q have the same tangents at the ends of the segment but different derivatives

Generate different Hermite curves

This techniques is used in drawing applications
Higher Dimensional Approximations

The techniques for both interpolating and Hermite curves can be used with higher dimensional parametric polynomials.

For interpolating form, the resulting matrix becomes increasingly more ill-conditioned and the resulting curves less smooth and more prone to numerical errors.

In both cases, there is more work in rendering the resulting polynomial curves and surfaces.
Beziers and B-Spline Curves and Surfaces

Beziere curves and surfaces
Derive the required matrices
Introduce the B-spline and compare it to the standard cubic Bezier
Bezier's Idea

In computer graphics and CAD, we do not usually have data about derivatives.

Bezier suggested using the same 4 data points as with the cubic interpolating curve to approximate the derivatives in the Hermite form.
Approximating Derivatives

\[ p'(0) \approx \frac{p_1 - p_0}{1/3} \]

\[ p'(1) \approx \frac{p_3 - p_2}{1/3} \]

- \( p_1 \) located at \( u = 1/3 \)
- \( p_2 \) located at \( u = 2/3 \)
- \( p_0 \)
- \( p_3 \)
Equations

Interpolating conditions are the same

\[ p(0) = p_0 = c_0 \]
\[ p(1) = p_3 = c_0 + c_1 + c_2 + c_3 \]

Approximating derivative conditions

\[ p'(0) = 3(p_1 - p_0) = c_0 \]
\[ p'(1) = 3(p_3 - p_2) = c_1 + 2c_2 + 3c_3 \]

Solve four linear equations for \( c = M_B p \)
Beziers Matrix

\[
M_B = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}
\]

\[p(u) = u^T M_B p = b(u)^T p\]  

blending functions
Blending Functions

\[ b(u) = \begin{bmatrix} (1 - u)^3 \\ 3u(1 - u)^2 \\ 3u^2(1 - u) \\ u^3 \end{bmatrix} \]

Note that all zeros are at 0 and 1 which forces the functions to be smooth over (0,1)
Bernstein Polynomials

The blending functions are a special case of the Bernstein polynomials:

$$b_{kd}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k}$$

These polynomials give the blending polynomials for any degree Bezier form. Their properties are:

- All zeros are located at 0 and 1
- For any degree, they all sum to 1
- Inside (0,1), they all evaluate to values between 0 and 1.
Convex Hull Property

The properties of the Bernstein polynomials ensure that all Bezier curves lie in the convex hull of their control points.

Hence, even though we do not interpolate all the data, we cannot be too far away.
Beziers Patches

Using same data array $\mathbf{P} = [p_{ij}]$ as with interpolating form

$$p(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) p_{ij} = u^T \mathbf{M}_B \mathbf{P} \mathbf{M}_B^T v$$

Patch lies in convex hull
Analysis

Although the Bezier form is much better than the interpolating form, we have the derivatives are not continuous at join points.

Can we do better?

A: Go to higher order Bezier
   More work
   Derivative continuity still only approximate

B: Apply different conditions
   Tricky without letting order increase
B-Splines

Basis splines: use the data at \( p = [p_{i-2} \ p_{i-1} \ p_i \ p_{i+1}]^T \) to define curve only between \( p_{i-1} \) and \( p_i \).

Allows us to apply more continuity conditions to each segment.

For cubics, we can have continuity of function, first and second derivatives at join points.

Cost is 3 times as much work for curves.

Add one new point each time rather than three.

For surfaces, we do 9 times as much work.
Cubic B-spline

\[ p(u) = u^T M_s p = b(u)^T p \]

\[
M_s = \begin{bmatrix}
1 & 4 & 1 & 0 \\
-3 & 0 & 3 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}
\]
Blending Functions

\[ b(u) = \frac{1}{6} \begin{bmatrix} (1-u)^3 \\ 4 - 6u^2 + 3u^3 \\ 1 + 3u + 3u^2 - 3u^2 \\ u^3 \end{bmatrix} \]

convex hull property
Extending to 3D ... B-Spline Patches

defined over only 1/9 of region

(we'll look at 3D B-Spline Patches again once we start working in 3D...)
Splines and Basis

If we examine the cubic B-spline from the perspective of each control (data) point, each interior point contributes (through the blending functions) to four segments.

We can rewrite \( p(u) \) in terms of the data points as

\[
p(u) = \sum B_i(u) p_i
\]

defining the basis functions \( \{B_i(u)\} \).
Basis Functions

In terms of the blending polynomials

\[ B_i(u) = \begin{cases} 
0 & u < i - 2 \\
 b_0(u + 2) & i - 2 \leq u < i - 1 \\
 b_1(u + 1) & i - 1 \leq u < i \\
 b_2(u) & i \leq u < i + 1 \\
 b_3(u - 1) & i + 1 \leq u < i + 2 \\
0 & u \geq i + 2 
\end{cases} \]