Geometric Theory, Algorithms, and Techniques

Hong Qin
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, New York 11794--4400
Tel: (631)632-8450; Fax: (631)632-8334
qin@cs.sunysb.edu
http://www.cs.sunysb.edu/~qin
Introduction

- Geometric modeling and visual computing
  - Computer graphics
    - Visualization, animation, virtual reality
  - CAD/CAM
    - Engineering, manufacturing
  - Computer vision
  - Physical simulation
  - Natural phenomena
3D Shape Representation

- Points (vertices), a set of points
- Lines, polylines, curve
- Triangles, polygons
- Triangular meshes, polygonal meshes
- Analytic (commonly-used) shape
- Quadric surfaces, sphere, ellipsoid, torus
- Superquadric surfaces, superellipse, superellipsoid
- Blobby models
Basic Shapes

- point
- line
- plane
- triangle
- polygon
- curve
- surface
- Curved solid
Fundamental Shapes

- Vertex (vertices)
- Line segments
- Triangle, triangular meshes
- Quadrilateral
- Polygon
- Curved object
- Tetrahedron, pyramid, hexahedron
- Many more…
Polygonal Meshes
Shaded Model
Mechanical Part
Subdivision model

Implicit model

NURBS model

PDE models
Building Structure
Mathematical Tools

• Parametric curves and surfaces
• Spline-based objects (piecewise polynomials)
• Explicit, implicit, and parametric representations
• The integrated way to look at the shape:
  – Object can be considered as a set of faces, each face can be further decomposed into a set of edges, each edge can be decomposed into vertices
• Subdivision models
• Other procedure-based models
• Sweeping
• Surfaces of revolution
• Volumetric models
Line Equation

- **Parametric representation**
  \[ l(p_0, p_1) = p_0 + (p_1 - p_0)u \]
  \[ u \in [0,1] \]

- **Parametric representation is not unique**

- **In general**
  \[ l(p_0, p_1) = 0.5(p_1 + p_0) + 0.5(p_1 - p_0)v \]
  \[ v \in [-1,1] \]

- **Re-parameterization (variable transformation)**

  \[
  \begin{align*}
  v &= (u - a) / (b - a) \\
  u &= (b - a) v + a \\
  q(v) &= p((b - a) v + a) \\
  v &\in [0,1]
  \end{align*}
  \]
Basic Concepts

- **Linear interpolation:**
  \[ \mathbf{v} = \mathbf{v}_0 (1 - t) + \mathbf{v}_1 (t) \]

- **Local coordinates:**
  \[ \mathbf{v} \in [\mathbf{v}_0, \mathbf{v}_1], t \in [0,1] \]

- **Reparameterization:**
  \[ f(u), u = g(v), f(g(v)) = h(v) \]

- **Affine transformation:**
  \[ f(ax + by) = af(x) + bf(y) \]
  \[ a + b = 1 \]

- **Polynomials**
- **Continuity**
Linear and Bilinear Interpolation

\[ p = (1 - u)a + ub \]

\[ e = (1 - u)a + uc \]
\[ f = (1 - u)b + ud \]
\[ p = (1 - v)e + vf \]
Fundamental Features

- **Geometry**
  - Position, direction, length, area, normal, tangent, etc.

- **Interaction**
  - Size, continuity, collision, intersection

- **Topology**

- **Differential properties**
  - Curvature, arc-length

- **Physical attributes**

- **Computer representation & data structure**

- **Others!**
Mathematical Formulations

- **Point:**
  \[ p = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \]

- **Line:**
  \[ l(u) = [a \ a \ a]^T u + [b \ b \ b]^T \]

- **Quadratic curve:**
  \[ q(u) = [a_x \ a_y \ a_z]^T u^2 + [b_x \ b_y \ b_z]^T u + [c_x \ c_y \ c_z]^T \]

- **Parametric domain and reparameterization:**
  \[ u \in [u_s, u_e]; \ \nu \in [0,1]; \ \nu = (u - u_s) / (u_e - u_s) \]
Parametric Polynomials

- High-order polynomials

\[
c(u) = \begin{bmatrix}
a_{0,x} \\
a_{0,y} \\
a_{0,z}
\end{bmatrix} + \ldots + \begin{bmatrix}
a_{i,x} \\
a_{i,y} \\
a_{i,z}
\end{bmatrix} u^i + \ldots + \begin{bmatrix}
a_{n,x} \\
a_{n,y} \\
a_{n,z}
\end{bmatrix} u^n
\]

- No intuitive insight for the curved shape
- Difficult for piecewise smooth curves
Parametric Polynomials
How to Define a Curve?

• Specify a set of points for interpolation and/or approximation with fixed or unfixed parameterization

\[
\begin{bmatrix}
x(u_i) \\
y(u_i) \\
z(u_i)
\end{bmatrix}
\]

\[
\begin{bmatrix}
x'(u_i) \\
y'(u_i) \\
z'(u_i)
\end{bmatrix}
\]

• Specify the derivatives at some locations
• What is the geometric meaning to specify derivatives?
• A set of constraints
• Solve constraint equations
One Example

- Two end-vertices: \( c(0) \) and \( c(1) \)
- One mid-point: \( c(0.5) \)
- Tangent at the mid-point: \( c'(0.5) \)
- Assuming 3D curve
Cubic Polynomials

- **Parametric representation (u is in [0,1])**
  
  \[
  \begin{bmatrix}
  x(u) \\
  y(u) \\
  z(u)
  \end{bmatrix}
  =
  \begin{bmatrix}
  a_3 \\
  b_3 \\
  c_3
  \end{bmatrix} u^3 +
  \begin{bmatrix}
  a_2 \\
  b_2 \\
  c_2
  \end{bmatrix} u^2 +
  \begin{bmatrix}
  a_1 \\
  b_1 \\
  c_1
  \end{bmatrix} u +
  \begin{bmatrix}
  a_0 \\
  b_0 \\
  c_0
  \end{bmatrix}
  \]

- **Each components are treated independently**
- **High-dimension curves can be easily defined**
- **Alternatively**
  
  \[
  x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}^T \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \end{bmatrix}
  = UA
  \]
  
  \[
  y(u) = UB
  \]
  
  \[
  z(u) = UC
  \]
Cubic Polynomial Example

- **Constraints:** two end-points, one mid-point, and tangent at the mid-point

\[
x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A
\]
\[
x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5^1 & 1 \end{bmatrix} A
\]
\[
x'(0.5) = \begin{bmatrix} 3(0.5)^2 & 2(0.5) & 1 & 0 \end{bmatrix} A
\]
\[
x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A
\]

- **In matrix form**

\[
\begin{bmatrix}
x(0) \\
x(0.5) \\
x'(0.5) \\
x(1)
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0.125 & 0.25 & 0.5 & 1 \\
0.75 & 1 & 1 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix} A
\]
Solve this Linear Equation

- Invert the matrix

\[
A = \begin{bmatrix}
-4 & 0 & -4 & 4 \\
8 & -4 & 6 & -4 \\
-5 & 4 & -2 & 1 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x(0) \\
x(0.5) \\
x'(0.5) \\
x(1)
\end{bmatrix}
\]

- Rewrite the curve expression

\[
x(u) = UM \begin{bmatrix} x(0) & x(0.5) & x'(0.5) & x(1) \end{bmatrix}^T
\]

\[
y(u) = UM \begin{bmatrix} y(0) & y(0.5) & y'(0.5) & y(1) \end{bmatrix}^T
\]

\[
z(u) = UM \begin{bmatrix} z(0) & z(0.5) & z'(0.5) & z(1) \end{bmatrix}^T
\]
Basis Functions

• **Special polynomials**

\[
\begin{align*}
 f_1(u) &= -4u^3 + 8u^2 - 5u + 1 \\
 f_2(u) &= -4u^2 + 4u \\
 f_3(u) &= -4u^3 + 6u^2 - 2u \\
 f_4(u) &= 4u^3 - 4u^2 + 1
\end{align*}
\]

• **What is the image of these basis functions?**

• **Polynomial curve can be defined by**

\[
c(u) = c(0)f_1(u) + c(0.5)f_2(u) + c'(0.5)f_3(u) + c(1)f_4(u)
\]

• **Observations**

  – More intuitive, easy to control, polynomials
Lagrange Curve

- Point interpolation
Lagrange Curves

- Curve
  \[ c(u) = \sum_{i=0}^{n} a_i L_i^n(u) \]

- Lagrange polynomials of degree \( n \):
  \[ L_i^n(u) \]

- Knot sequence:
  \[ u_0, \ldots, u_n \]

- Kronecker delta:
  \[ L_i^n(u_j) = \delta_{ij} \]

- The curve interpolate all the data point, but unwanted oscillation
Lagrange Basis Functions

\[ L^n_i(u_j) = \begin{cases} 
1 & i = j (i, j = 0, 1, \ldots, n) \\
0 & \text{Otherwise}
\end{cases} \]

\[ L^n_0(u) = \frac{(u - u_1)(u - u_2)\ldots(u - u_n)}{(u_0 - u_1)(u_0 - u_2)\ldots(u_0 - u_n)} \]

\[ L^n_i(u) = \frac{(u - u_0)\ldots(u - u_{i-1})(u - u_{i+1})\ldots(u - u_n)}{(u_i - u_0)\ldots(u_i - u_{i-1})(u_i - u_{i+1})\ldots(u_i - u_n)} \]

\[ L^n_n(u) = \frac{(u - u_0)\ldots(u - u_{n-2})(u - u_{n-1})}{(u_n - u_0)\ldots(u_n - u_{n-2})(u_n - u_{n-1})} \]
Cubic Hermite Splines
Cubic Hermite Curve

- **Hermite curve**
  \[
  c(u) = \begin{bmatrix}
  x(u) \\
  y(u) \\
  z(u)
  \end{bmatrix}
  \]

- **Two end-points and two tangents at end-points**
  \[
  \begin{bmatrix}
  x(0) \\
  x(1) \\
  x'(0) \\
  x'(1)
  \end{bmatrix}
  =
  \begin{bmatrix}
  0 & 0 & 0 & 1 \\
  1 & 1 & 1 & 1 \\
  0 & 0 & 1 & 0 \\
  3 & 2 & 1 & 0
  \end{bmatrix}
  A
  \]

- **Matrix inversion**
  \[
  x(u) = U \begin{bmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0
  \end{bmatrix}
  \begin{bmatrix}
  x(0) \\
  x(1) \\
  x'(0) \\
  x'(1)
  \end{bmatrix}
  \]
  \[
  y(u) = UM \begin{bmatrix}
  y(0) \\
  y(1) \\
  y'(0) \\
  y'(1)
  \end{bmatrix}^T
  \]
  \[
  z(u) = UM \begin{bmatrix}
  z(0) \\
  z(1) \\
  z'(0) \\
  z'(1)
  \end{bmatrix}^T
  \]
Hermite Curve

• **Basis functions**

\[
\begin{align*}
    f_1(u) &= 2u^3 - 3u^2 + 1 \\
    f_2(u) &= -2u^3 + 3u^2 \\
    f_3(u) &= u^3 - 2u^2 + u \\
    f_4(u) &= u^3 - u^2
\end{align*}
\]

• **Display the image of these basis functions and the Hermite curve itself**

\[
\mathbf{c}(u) = \mathbf{c}(0)f_1(u) + \mathbf{c}(1)f_2(u) + \mathbf{c}'(0)f_3(u) + \mathbf{c}'(1)f_4(u)
\]
Cubic Hermite Splines

- **Two vertices and two tangent vectors:**
  \[ c(0) = v_0, \; c(1) = v_1; \]
  \[ c^{(1)}(0) = d_0, \; c^{(1)}(1) = d_1; \]

- **Hermite curve**
  \[ c(u) = v_0H_0^3(u) + v_1H_1^3(u) + d_0H_2^3(u) + d_1H_3^3(u); \]
  \[ H_0^3(u) = f_1(u), \; H_1^3(u) = f_2(u), \; H_2^3(u) = f_3(u), \; H_3^3(u) = f_4(u) \]
Hermite Splines

• Higher-order polynomials

\[ \mathbf{c}(u) = v_0^0 H_0^n(u) + v_0^1 H_1^n(u) + \ldots + v_0^{(n-1)/2} H_{(n-1)/2}^n(u) + v_1^{(n-1)/2} H_{(n+1)/2}^n(u) + \ldots + v_1^1 H_{(n-1)}^n(u) + v_1^0 H_n^n(u); \]

\[ v_i^0 = \mathbf{c}^{(i)}(0), \quad v_i^1 = \mathbf{c}^{(i)}(1), \quad i = 0, \ldots, (n-1)/2; \]

• Note that, \( n \) is odd!

• Geometric intuition

• Higher-order derivatives are required
Why Cubic Polynomials

- Lowest degree for specifying curve in space
- Lowest degree for specifying points to interpolate and tangents to interpolate
- Commonly used in computer graphics
- Lower degree has too little flexibility
- Higher degree is unnecessarily complex, exhibit undesired wiggles
Variations of Hermite Curve

- Variations of Hermite curves
  \[ p_0 = c(0) \]
  \[ p_3 = c(1) \]
  \[ c'(0) = 3(p_1 - p_0), p_1 = p_0 + c'(0)/3 \]
  \[ c'(1) = 3(p_3 - p_2), p_2 = p_3 - c'(1)/3 \]

- In matrix form (x-component only)

\[
\begin{bmatrix}
  c(0)_x \\
  c(1)_x \\
  c'(0)_x \\
  c'(1)_x \\
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  -3 & 3 & 0 & 0 \\
  0 & 0 & -3 & 3 \\
\end{bmatrix}
\begin{bmatrix}
  p_{0,x} \\
  p_{0,x} \\
  p_{0,x} \\
  p_{0,x} \\
\end{bmatrix}
\]
Cubic Bezier Curves

- Four control points
- Curve geometry
Curve Mathematics (Cubic)

• Bezier curve
  \[ c(u) = \sum_{i=0}^{3} p_i B_i^3(u) \]

• Control points and basis functions
  \[
  B_0^3(u) = (1 - u)^3 \\
  B_1^3(u) = 3u(1 - u)^2 \\
  B_2^3(u) = 3u^2(1 - u) \\
  B_3^3(u) = u^3
  \]

• Image and properties of basis functions
Recursive Evaluation

- Recursive linear interpolation

\[
\begin{align*}
(1 - u) \quad (u) \\
p_0^0 & \quad p_1^0 & \quad p_2^0 & \quad p_3^0 \\
p_0^1 & \quad p_1^1 & \quad p_2^1 \\
p_0^2 & \quad p_1^2 \\
p_0^3 = c(u)
\end{align*}
\]
Recursive Subdivision Algorithm
Basic Properties (Cubic)

- The curve passes through the first and the last points (end-point interpolation)
- Linear combination of control points and basis functions
- Basis functions are all polynomials
- Basis functions sum to one (partition of unity)
- All basis functions are non-negative
- Convex hull (both necessary and sufficient)
- Predictability
Derivatives

- Tangent vectors can easily be evaluated at the end-points:
  \[ c'(0) = 3(p_1 - p_0); c'(1) = (p_3 - p_2) \]

- Second derivatives at end-points can also be easily computed:
  \[
  c^{(2)}(0) = 2 \times 3((p_2 - p_1) - (p_1 - p_0)) = 6(p_2 - 2p_1 + p_0) \\
  c^{(2)}(1) = 2 \times 3((p_3 - p_2) - (p_2 - p_1)) = 6(p_3 - 2p_2 + p_1)
  \]
Derivative Curve

• The derivative of a cubic Bezier curve is a quadratic Bezier curve

\[ c'(u) = -3(1-u)^2 p_0 + 3((1-u)^2 - 2u(1-u))p_1 + 3(2u(1-u) - u^2)p_2 + 3u^2 p_3 = 3(p_1 - p_0)(1-u)^2 + 3(p_2 - p_1)2u(1-u) + 3(p_3 - p_2)u^2 \]
More Properties (Cubic)

• Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

\[
\begin{align*}
\mathbf{c}(v), & \quad v \in [0, u] \\
\mathbf{c}(v), & \quad v \in [u, 1] \\
\mathbf{c}_l(u), & \quad u \in [0, 1] \\
\mathbf{c}_r(u), & \quad u \in [0, 1]
\end{align*}
\]

• The control points for the left and the right are

\[
\begin{bmatrix}
p_0 & p_1 & p_2 & p_3 \\
p_0 & p_0 & p_0 & p_0 \\
p_3 & p_2 & p_1 & p_0 \\
p_0 & p_1 & p_2 & p_3
\end{bmatrix}
\]
High-Degree Curves

- Generalizing to high-degree curves

\[
\begin{bmatrix}
  x(u) \\
  y(u) \\
  z(u)
\end{bmatrix} = \sum_{i=0}^{n} \begin{bmatrix}
  a_i \\
  b_i \\
  c_i
\end{bmatrix} u^i
\]

- Advantages:
  - Easy to compute, Infinitely differentiable

- Disadvantages:
  - Computationally complex, undulation, undesired wiggles

- How about high-order Hermite? Not natural!!!
Beziers Splines

-Bezier curves of degree $n$

$$c(u) = \sum_{i=0}^{n} p_i B_i^n(u)$$

-Control points and basis functions (Bernstein polynomials of degree $n$):

$$B_i^n(u) = \binom{n}{i} (1-u)^{n-i} u^i$$

$$\binom{n}{i} = \frac{n!}{(n-i)!i!}$$
Recursive Computation

\[ p_i^0 = p_i, \quad i = 0, 1, 2, \ldots n \]

\[ p_i^j = (1 - u)p_i^{j-1} + up_i^{j-1} \]

\[ c(u) = p_0^n(u) \]
Recursive Computation

• **N+1 levels**

\[
\begin{pmatrix}
1 & -u \\
p_{0} & \cdots & \cdots & p_{1}^{n} \\
p_{1} & \cdots & \cdots & p_{1}^{n-1} \\
\vdots & \ddots & \ddots & \vdots \\
p_{n-1} & \cdots & \cdots & p_{1}^{n-1} \\
p_{0} & \cdots & \cdots & p_{1}^{n-1} \\
p_{0}^{n} & = & c & (u)
\end{pmatrix}
\]
Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $c(0) = p_0$, $c(1) = p_n$
- Binomial expansion theorem

\[
((1 - u) + u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1 - u)^{n-i}
\]

- Convex hull: the curve is bounded by the convex hull defined by control points
More Properties

- Recursive subdivision and evaluation
- Symmetry: \( c(u) \) and \( c(1-u) \) are defined by the same set of points, but different ordering

\[
p_0, \ldots, p_n; \\
p_n, \ldots, p_0
\]
Tangents and Derivatives

• **End-point tangents:**
  
  \[
  c'(0) = n(p_1 - p_0) \\
  c'(1) = n(p_n - p_{n-1})
  \]

• **I-th derivatives at two end-points depend on**
  
  \[
  p_0, ..., p_i; \\
  p_n, ..., p_{n-i}
  \]

• **Derivatives at non-end-points involve all control points**
Other Advanced Topics

- Efficient evaluation algorithm
- Differentiation and integration
- Degree elevation
  - Use a polynomial of degree \( (n+1) \) to express that of degree \( n \)
- Composite curves
- Geometric continuity
- Display of curve
Bezier Curve Rendering

- Use its control polygon to approximate the curve
- Recursive subdivision till the tolerance is satisfied
- Algorithm go here
  - If the current control polygon is flat (with tolerance), then output the line segments, else subdivide the curve at $u=0.5$
  - Compute control points for the left half and the right half, respectively
  - Recursively call the same procedure for the left one and the right one
High-Degree Polynomials

- More degrees of freedom
- Easy to compute
- Infinitely differentiable
- **Drawbacks:**
  - High-order
  - Global control
  - Expensive to compute, complex
  - undulation
Piecewise Polynomials

- Piecewise --- different polynomials for different parts of the curve
- Advantages --- flexible, low-degree
- Disadvantages --- how to ensure smoothness at the joints (continuity)
Piecewise Curves
Piecewise Bezier Curves
Continuity

• One of the fundamental concepts

• Commonly used cases: \( C^0, C^1, C^2 \)

• Consider two curves: \( a(u) \) and \( b(u) \) (\( u \) is in \([0,1]\))
Positional Continuity

\[ a(1) = b(0) \]
Derivative Continuity

\[ a(1) = b(0) \]
\[ a'(1) = b'(0) \]
General Continuity

- Cn continuity: derivatives (up to n-th) are the same at the joining point
  \[ a^{(i)}(1) = b^{(i)}(0) \]
  \[ i = 0, 1, 2, ..., n \]

- The prior definition is for parametric continuity
- Parametric continuity depends on parameterization! But, parameterization is not unique!
- Different parametric representations may express the same geometry
- Re-parameterization can be easily implemented
- Another type of continuity: geometric continuity, or G0
Geometric Continuity

- \(G_0\) and \(G_1\)
Geometric Continuity

- Depend on the curve geometry
- DO NOT depend on the underlying parameterization
- G0: the same joint
- G1: two curve tangents at the joint align, but may (or may not) have the same magnitude
- G1: it is C1 after the reparameterization
- Which condition is stronger???
- Examples
Piecwise Hermite Curves

- How to build an interactive system to satisfy various constraints
- $C^0$ continuity
  \[ a(1) = b(0) \]
- $C^1$ continuity
  \[ a'(1) = b'(0) \]
- $G^1$ continuity
  \[ a(1) = b(0) \]
  \[ a'(1) = \alpha b'(0) \]
Piecewise Hermite Curves
Piecewise Bezier Curves
Piecewise Bezier Curves

- **C0 continuity**
  \[ p_3 = q_0 \]
- **C1 continuity**
  \[ p_3 = q_0 \]
  \[ (p_3 - p_2) = (q_1 - q_0) \]
- **G1 continuity**
  \[ p_3 = q_0 \]
  \[ (p_3 - p_2) = \alpha(q_1 - q_0) \]
- **C2 continuity**
  \[ p_3 = q_0 \]
  \[ (p_3 - p_2) = (q_1 - q_0) \]
  \[ p_3 - 2p_2 + p_1 = q_2 - 2q_1 + q_0 \]
- **Geometric interpretation**
- **G2 continuity**
Piecewise C2 Bezier Curves
Continuity Summary

- **C0**: straightforward, but not enough
- **C3**: too constrained
- Piecewise curves with Hermite and Bezier representations satisfying various continuity conditions
- Interactive system for C2 interpolating splines using piecewise Bezier curves
- Advantages and disadvantages
C2 Interpolating Splines
Natural C2 Cubic Splines

- A set of piecewise cubic polynomials

\[
\mathbf{c}_i(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}
\]

- C2 continuity at each vertex
Natural C2 Cubic Splines
Natural Splines

- Interpolate all control points
- Equivalent to a thin strip of metal in a physical sense
- Forced to pass through a set of desired points
- No local control (global control)
- \( N+1 \) control points
- \( N \) pieces
- \( 2(n-1) \) conditions
- We need two additional conditions
Natural Splines

• **Interactive design system**
  – Specify derivatives at two end-points
  – Specify the two internal control points that define the first curve span
  – Natural end conditions: second-order derivatives at two end points are defined to be zero

• **Advantages:** interpolation, C2

• **Disadvantages:** no local control (if one point is changed, the entire curve will move)

• **How to overcome this drawback:** B-Splines
B-Splines Motivation

- The goal is local control!!!
- B-splines provide local control
- Do not interpolate control points
- C2 continuity
- Alternatively
  - Catmull-Rom Splines
  - Keep interpolations
  - Give up C2 continuity (only C1 is achieved)
  - Will be discussed later!!!
C2 Approximating Splines
From B-Splines to Bezier
Uniform B-Splines

- **B-spline control points:** \( p_0, p_1, \ldots, p_n \)
- **Piecewise Bezier curves with C2 continuity at joints**
- **Bezier control points:**

\[
\begin{align*}
\mathbf{v}_0 &= \mathbf{p}_0 \\
\mathbf{v}_1 &= \frac{2\mathbf{p}_1 + \mathbf{p}_2}{3} \\
\mathbf{v}_2 &= \frac{\mathbf{p}_1 + 2\mathbf{p}_2}{3} \\
\mathbf{v}_3 &= \frac{1}{6} (\mathbf{p}_0 + 4\mathbf{p}_1 + \mathbf{p}_2) \\
\mathbf{v}_4 &= \frac{1}{6} (\mathbf{p}_1 + 4\mathbf{p}_2 + \mathbf{p}_3)
\end{align*}
\]
Uniform B-Splines

- In general, I-th segment of B-splines is determined by four consecutive B-spline control points

\[
\begin{align*}
v_1 &= \frac{2}{3} p_{i+1} + \frac{1}{3} p_{i+2} \\
v_2 &= \frac{1}{3} p_{i+1} + \frac{2}{3} p_{i+2} \\
v_0 &= \frac{1}{6} (p_i + 4p_{i+1} + p_{i+2}) \\
v_3 &= \frac{1}{6} (p_{i+1} + 4p_{i+2} + p_{i+3})
\end{align*}
\]
Uniform B-Splines

- In matrix form

\[
\begin{bmatrix}
  v_0 \\
  v_1 \\
  v_2 \\
  v_3
\end{bmatrix} = \frac{1}{6}
\begin{bmatrix}
  1 & 4 & 1 & 0 \\
  0 & 4 & 2 & 0 \\
  0 & 2 & 4 & 0 \\
  0 & 1 & 4 & 1
\end{bmatrix}
\begin{bmatrix}
  p_i \\
  p_{i+1} \\
  p_{i+2} \\
  p_{i+3}
\end{bmatrix}
\]

- Question: how many Bezier segments???
B-Spline Properties

- C2 continuity, Approximation, Local control, convex hull
- Each segment is determined by four control points
- Questions: what happens if we put more than one control points in the same location???
  - Double vertices, triple vertices, collinear vertices
- End conditions
  - Double endpoints: curve will be tangent to line between first distinct points
  - Triple endpoint: curve interpolate endpoint, start with a line segment
- B-spline display: transform it to Bezier curves
Catmull-Rom Splines
Catmull-Rom Splines

- Keep interpolation
- Give up C2 continuity
- Control tangents locally
- Idea: Bezier curve between successive points
- How to determine two internal vertices

\[
\begin{align*}
    c(0) &= p_i = v_0, \quad c(1) = p_{i+1} = v_3 \\
    c'(0) &= \frac{p_{i+1} - p_{i-1}}{2} = 3(v_1 - v_0) \\
    c'(1) &= \frac{p_{i+2} - p_i}{2} = 3(v_3 - v_2) \\
    v_1 &= \frac{p_{i+1} + 6p_i - p_{i-1}}{6} \\
    v_2 &= \frac{-p_{i+2} + 6p_{i+1} + p_i}{6}
\end{align*}
\]
Catmull-Rom Splines

- **In matrix form**

\[
\begin{bmatrix}
v_0 \\
v_1 \\
v_2 \\
v_3 \\
\end{bmatrix} =
\begin{bmatrix}
0 & 6 & 0 & 0 \\
1 & -1 & 6 & 1 & 0 \\
6 & 0 & 1 & 6 & -1 \\
0 & 0 & 6 & 0 \\
\end{bmatrix}
\begin{bmatrix}
p_{i-1} \\
p_i \\
p_{i+1} \\
p_{i+2} \\
\end{bmatrix}
\]

- **Problem:** boundary conditions

- **Properties:** C1, interpolation, local control, non-convex-hull
Cardinal Splines

- Four vertices define end-points and their associated tangents

\[
\begin{align*}
c(0) &= v_1, c(1) = v_2 \\
c^{(1)}(0) &= \frac{1}{2} (1 - \alpha)(v_2 - v_0) \\
c^{(1)}(1) &= \frac{1}{2} (1 - \alpha)(v_3 - v_1)
\end{align*}
\]

- Special case: Catmull-Rom splines when \(\alpha = 0\)
- More general case: Kochanek-Bartels splines
  - Tension, bias, continuity parameters
Cardinal Splines
Kochanek-Bartels Splines

• Four vertices to define four conditions

\[
\begin{align*}
\mathbf{c}(0) &= \mathbf{v}_1, \mathbf{c}(1) = \mathbf{v}_2 \\
\mathbf{c}^{(1)}(0) &= \frac{1}{2} (1 - \alpha)((1 + \beta)(1 - \gamma)(\mathbf{v}_1 - \mathbf{v}_0) + (1 - \beta)(1 + \gamma)(\mathbf{v}_2 - \mathbf{v}_1)) \\
\mathbf{c}^{(1)}(1) &= \frac{1}{2} (1 - \alpha)((1 + \beta)(1 + \gamma)(\mathbf{v}_2 - \mathbf{v}_1) + (1 - \beta)(1 - \gamma)(\mathbf{v}_3 - \mathbf{v}_2))
\end{align*}
\]

– Tension parameter: \( \alpha \)
– Bias parameter: \( \beta \)
– Continuity parameter: \( \gamma \)
Piecewise B-Splines
B-Spline Basis Functions

\[ B_{i,1}(u) = \begin{cases} 
1 & u_i \leq u < u_{i+1} \\
0 & otherwise 
\end{cases} \]

\[ B_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} B_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} B_{i+1,k-1}(u) \]
Basis Functions

• **Linear examples**

\[
B_{0,2}(u) = \begin{cases} 
  u & u \in [0,1] \\
  2 - u & u \in [1,2] 
\end{cases}
\]

\[
B_{1,2}(u) = \begin{cases} 
  u - 1 & u \in [1,2] \\
  3 - u & u \in [2,3] 
\end{cases}
\]

\[
B_{2,2}(u) = \begin{cases} 
  u - 2 & u \in [2,3] \\
  4 - u & u \in [3,4] 
\end{cases}
\]

• **How does it look like???**
Basis Functions

• **Quadratic cases (knot vector is \([0,1,2,3,4,5,6]\))**

\[
B_{0,3}(u) = \begin{cases} 
\frac{1}{2} u^2, & 0 \leq u < 1 \\
\frac{1}{2} u ( 2 - u ) + \frac{1}{2} (u - 1)( 3 - u ), & 1 \leq u < 2 \\
\frac{1}{2} (3 - u)^2, & 2 \leq u < 3 
\end{cases}
\]

\[
B_{1,3}(u) = \begin{cases} 
\frac{1}{2} (u - 1)^2, & 1 \leq u < 2 \\
\frac{1}{2} (u - 1)( 3 - u ) + \frac{1}{2} (u - 2)( 4 - u ), & 2 \leq u < 3 \\
\frac{1}{2} (4 - u)^2, & 3 \leq u < 4 
\end{cases}
\]

\[
B_{2,3}(u) = \ldots \\
B_{3,3}(u) = \ldots
\]

• **Cubic example**
B-Spline Basis Function Image
B-Splines

- **Mathematics**
  \[ c(u) = \sum_{i=0}^{n} p_i B_{i,k}(u) \]
- **Control points and basis functions of degree (k-1)**
- **Piecewise polynomials**
- **Basis functions are defined recursively**
- **We also have to introduce a knot sequence (n+k+1) in a non-decreasing order**
  \[ u_0, u_1, u_2, u_3, \ldots, u_{n+k} \]
- **Note that, the parametric domain:** \[ u \in [u_{k-1}, u_{n+1}] \]
Basis Functions

\[ B_{0,1} \quad B_{1,1} \quad B_{2,1} \quad B_{3,1} \quad B_{4,1} \quad B_{5,1} \quad B_{6,1} \]

\[ B_{0,2} \quad B_{1,2} \quad B_{2,2} \quad B_{3,2} \quad B_{4,2} \quad B_{5,2} \]

\[ B_{0,3} \quad B_{1,3} \quad B_{2,3} \quad B_{3,3} \quad B_{4,3} \]

\[ B_{0,4} \quad B_{1,4} \quad B_{2,4} \quad B_{3,4} \]
B-Spline Facts

• The curve is a linear combination of control points and their associated basis functions \((n+1)\) control points and basis functions, respectively.

• Basis functions are piecewise polynomials defined recursively over a set of non-decreasing knots

\[ \{u_0, \ldots, u_{k-1}, \ldots, u_{n+1}, \ldots, u_{n+k}\} \]

• The degree of basis functions is independent of the number of control points (note that, \(I\) is index, \(k\) is the order, \(k-1\) is the degree).

• The first \(k\) and last \(k\) knots do NOT contribute to the parametric domain. Parametric domain is only defined by a subset of knots.
B-Spline Properties

• $C(u)$: piecewise polynomial of degree $(k-1)$
• Continuity at joints: $C(k-2)$
• The number of control points and basis functions: $(n+1)$
• One typical basis function is defined over $k$ sub-intervals which are specified by $k+1$ knots $([u(k), u(I+k)])$
• There are $n+k+1$ knots in total, knot sequence divides the parametric axis into $n+k$ sub-intervals
• There are $(n+1)-(k-1)=n-k+2$ sub-intervals within the parametric domain $([u(k-1), u(n+1)])$
B-Spline Properties

- There are $n-k+2$ piecewise polynomials
- Each curve span is influenced by $k$ control points
- Each control point at most affects $k$ curve spans
- Local control!!!
- Convex hull
- The degree of B-spline polynomial can be independent from the number of control points
- Compare B-spline with Bezier!!!
- Key components: control points, basis functions, knots, parametric domain, local vs. global control, continuity
B-Spline Properties

- Partition of unity, positivity, and recursive evaluation of basis functions
- Special cases: Bezier splines
- Efficient algorithms and tools
  - Evaluation, knot insertion, degree elevation, derivative, integration, continuity
- Composite Bezier curves for B-splines
Uniform B-Spline
Another Formulation

- **Uniform B-spline**
- **Parameter normalization (u is in \([0,1]\))**
- **End-point positions and tangents**

\[
\begin{align*}
  c(0) &= \frac{1}{6} (p_0 + 4p_1 + p_2) \\
  c(1) &= \frac{1}{6} (p_1 + 4p_2 + p_3) \\
  c'(0) &= \frac{1}{2} (p_2 - p_0) \\
  c'(1) &= \frac{1}{2} (p_3 - p_1)
\end{align*}
\]
Another Formulation

- **Matrix representation**
  \[
  \begin{bmatrix}
  c(0) \\
  c(1) \\
  c'(0) \\
  c'(1)
  \end{bmatrix}
  = UM_h M' \begin{bmatrix}
  p_0 \\
  p_1 \\
  p_2 \\
  p_3
  \end{bmatrix} = UM p
  \]

- **Basis matrix**
  \[
  M = \frac{1}{6}
  \begin{bmatrix}
  -1 & 3 & -3 & 1 \\
  3 & -6 & 3 & 0 \\
  -3 & 0 & 3 & 0 \\
  1 & 4 & 1 & 0
  \end{bmatrix}
  \]
Basis Functions

- Note that, \( u \) is now in \([0,1]\)

\[
B_{0,4}(u) = \frac{1}{6}(1 - u)^3
\]

\[
B_{1,4}(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)
\]

\[
B_{2,4}(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)
\]

\[
B_{3,4}(u) = \frac{1}{6}(u)^3
\]
### B-Spline Rendering

- **Transform it to a set of Bezier curves**
- **Convert the I-th span into a Bezier representation**

\[
\begin{align*}
&\mathbf{p}_i, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_{i+k-1} \\
&\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1}
\end{align*}
\]

- **Consider the entire B-spline curve**

\[
\begin{align*}
&\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n \\
&\mathbf{v}_0, \ldots, \mathbf{v}_3, \mathbf{v}_4, \ldots, \mathbf{v}_7, \ldots, \mathbf{v}_{4(n-3)}, \ldots, \mathbf{v}_{4(n-3)+3}
\end{align*}
\]
Matrix Expression

\[
\begin{bmatrix}
 v_0 \\
 M \\
 v_{4(n-3)+3}
\end{bmatrix}
= \begin{bmatrix}
 p_0 \\
 M \\
 p_n
\end{bmatrix}
\]

- The matrix structure and components of B?

\[ q = Av = AB \]

- The matrix structure and components of A?
B-Spline Discretization

- Parametric domain: \([u(k-1), u(n+1)]\)
- There are \(n+2-k\) curve spans (pieces)
- Assuming \(m+1\) points per span (uniform sampling)
- Total sampling points \(m(n+2-k)+1=l\)
- B-spline discretization with corresponding parametric values:

\[
q_0, \ldots, q_{l-1} \\
v_0, \ldots, v_{l-1} \\
q_i = c(v_i) = \sum_{j=0}^{n} p_j B_{j,k}(v_i)
\]
B-Spline Discretization

- **Matrix equation**

\[
\begin{bmatrix}
q_0 \\
M \\
q_{l-1}
\end{bmatrix} = 
\begin{bmatrix}
B_{0,k}(v_0) & \Lambda & B_{n,k}(v_0) \\
M & O & M \\
B_{0,k}(v_{l-1}) & \Lambda & B_{n,k}(v_{l-1})
\end{bmatrix}
\begin{bmatrix}
p_0 \\
M \\
p_n
\end{bmatrix}
\]

- **A is** \((l)\times(n+1)\) **matrix, in general** \((l)\) **is much larger than** \((n+1)\), so **A is sparse**

- **The linear discretization for both modeling and rendering**
From B-Splines to NURBS

• What are NURBS???
• Non Uniform Rational B-Splines (NURBS)
• Rational curve motivation
• Polynomial-based splines can not represent commonly-used analytic shapes such as conic sections (e.g., circles, ellipses, parabolas)
• Rational splines can achieve this goal
• NURBS are a unified representation
  – Polynomial, conic section, etc.
  – Industry standard
From B-Splines to NURBS

• B-splines

\[ c(u) = \sum_{i=0}^{n} \begin{bmatrix} p_{i,x} w_i \\ p_{i,y} w_i \\ p_{i,z} w_i \\ w_i \end{bmatrix} B_{i,k}(u) \]

• NURBS (curve)

\[ c(u) = \frac{\sum_{i=0}^{n} p_{i} w_{i} B_{i,k}(u)}{\sum_{i=0}^{n} w_{i} B_{i,k}(u)} \]
Geometric NURBS

- Non-Uniform Rational B-Splines
- CAGD industry standard --- useful properties
- Degrees of freedom
  - Control points
  - Weights
Rational Bezier Curve

- Projecting a Bezier curve onto w=1 plane
From B-Splines to NURBS
NURBS Weights

- Weight increase “attracts” the curve towards the associated control point
- Weight decrease “pushes away” the curve from the associated control point
NURBS for Analytic Shapes

• Conic sections
• Natural quadrics
• Extruded surfaces
• Ruled surfaces
• Surfaces of revolution
NURBS Circle

\[ a, b, c, d, e, d, g \]

\[ w_i = 1, 0.5, 0.5, 1, 0.5, 0.5, 1 \]

\[ \text{knot} = [0, 0, 0, 1, 2, 2, 3, 4, 4, 4] \]
NURBS Curve

- **Geometric components**
  - Control points, parametric domain, weights, knots

- **Homogeneous representation of B-splines**

- **Geometric meaning --- obtained from projection**

- **Properties of NURBS**
  - Represent standard shapes, invariant under perspective projection, B-spline is a special case, weights as extra degrees of freedom, common analytic shapes such as circles, clear geometric meaning of weights
NURBS Properties

- Generalization of B-splines and Bezier splines
- Unified formulation for free-form and analytic shape
- Weights as extra DOFs
- Various smoothness requirements
- Powerful geometric toolkits
- Efficient and fast evaluation algorithm
- Invariance under standard transformations
- Composite curves
- Continuity conditions
Geometric Modeling

- Why geometric modeling
- Fundamental for visual computing
  - Graphics, visualization
  - Computer aided design and manufacturing
  - Imaging
  - Entertainment, etc.
- Critical for virtual engineering
- Interaction
- Geometric information for decision making
Parameterization
Surfaces

- **From curves to surfaces**
- **A simple curve example (Bezier)**

\[ c(u) = \sum_{i=0}^{3} p_i B_i(u) \quad u \in [0,1] \]

- **Consider each control point now becoming a Bezier curve**

\[ p_i = \sum_{j=0}^{3} p_{i,j} B_j(v) \quad v \in [0,1] \]
Surfaces

- Then, we have
- Matrix form

\[ s(u, v) = \sum_{i=0}^{3} \left( \sum_{j=0}^{3} p_{i,j} B_j(v) \right) B(u) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} B_i(u) B_j(v) \]

\[ s(u, v) = \begin{bmatrix} B_0(u) & B_1(u) & B_2(u) & B_3(u) \end{bmatrix} \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,0} & p_{2,1} & p_{2,2} & p_{2,3} \\ p_{3,0} & p_{3,1} & p_{3,2} & p_{3,3} \end{bmatrix} = \begin{bmatrix} B_0(u) \\ B_1(u) \\ B_2(u) \\ B_3(u) \end{bmatrix} = UMPM^T V^T \]
Surfaces

- Further generalize to degree of n and m along two parametric directions

$$s(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{i,j} B_i^n(u) B_j^m(v)$$

- Question: which control points are interpolated?
- How about B-spline surfaces???
Tensor Product Surfaces

- **Where are they from?**
- **Monomial form**
  \[ s(u, v) = \sum_i \sum_j a_{i,j} u^i v^j \]
- **Bezier surface**
  \[ s(u, v) = \sum_i \sum_j p_{i,j} B_i^m(u) B_j^n(v) \]
- **B-spline surface**
  \[ s(u, v) = \sum_{i=0}^m \sum_{j=0}^n p_{i,j} B_{i,k}(u) B_{j,l}(v) \]
- **General case**
  \[ s(u, v) = \sum_i \sum_j v_{i,j} F_i(u) G_j(v) \]
Tensor Product Surface

- Bezier Surface
B-Splines

- B-spline curves
  \[ c(u) = \sum_{i=0}^{n} p_i B_{i,k}(u) \]

- Tensor product B-splines
  \[ s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} B_{i,k}(u)B_{j,l}(v) \]

- Question again: which control points are interpolated???
- Another question: can we get NURBS surface this way???
- Answer: NO!!! NURBS are not tensor-product surfaces
- Another question: can we have NURBS surface?
- YES!!!
NURBS Surface

- **NURBS surface mathematics**

\[
\mathbf{s}(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{p}_{i,j} w_{i,j} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} B_{i,k}(u) B_{j,l}(v)}
\]

- Understand this geometric construction

- Question: why is it not the tensor-product formulation?? Compare it with Bezier and B-spline construction
NURBS Surface

- **Parametric variables**: \( u \) and \( v \)
- **Control points and their associated weights**: \((m+1)(n+1)\)
- **Degrees of basis functions**: \((k-1)\) and \((l-1)\)
- **Knot sequence**: 

  \[
  u_0 <= u_1 <= ... <= u_{m+k} \\
  v_0 <= v_1 <= ... <= v_{n+l}
  \]

- **Parametric domain**: 

  \[
  u_{k-1} <= u <= u_{m+1} \\
  v_{l-1} <= v <= v_{n+1}
  \]
NURBS Surface

- The same principle to generate curves via projection
- Idea: associate weights with control points
- Generalization of B-spline surface
Hermite Surfaces

• How about Hermite surfaces???
• Hermite Curve:

\[
c(u) = \begin{bmatrix} H_0(u) & H_1(u) & H_2(u) & H_3(u) \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \\ c'(0) \\ c'(1) \end{bmatrix}
\]

• \( C(0) \) is not a curve \( s(0,v) \) which is also a Hermite Curve:

\[
s(0,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix} \begin{bmatrix} s(0,0) \\ s(0,1) \\ s_v(0,0) \\ s_v(0,1) \end{bmatrix}
\]
Hermite Surfaces

• Similarly, \( c(1) \) is now a curve \( s(1,v) \) which is also a Hermite curve:

\[
s(1,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix}
\]

\[
\begin{bmatrix}
  s(1,0) \\
  s(1,1) \\
  s_v(1,0) \\
  s_v(1,1)
\end{bmatrix}
\]

• The same are for \( c'(0) \) and \( c'(1) \):

\[
s_u(0,v) = H(v)
\]

\[
s_u(1,v) = H(v)
\]
Hermite Surfaces

• It is time to put them together!

\[
s(u, v) = H(u) \begin{bmatrix}
  s(0,0) & s(0,1) & s_v(0,0) & s_v(0,1) \\
  s(1,0) & s(1,1) & s_v(1,0) & s_v(1,1) \\
  s_u(0,0) & s_u(0,1) & s_{uv}(0,0) & s_{uv}(0,1) \\
  s_u(1,0) & s_u(1,1) & s_{uv}(1,0) & s_{uv}(1,1)
\end{bmatrix} H(v)^T
\]

• Continuity conditions for surfaces
• Bezier surfaces, B-splines, NURBS, Hermite surfaces
• C1 and G1 continuity
Hermite Surfaces
Surface Normal
Surface Rendering

- Parametric grids \([0,1] \times [0,1]\) as a set of rectangles
Surface (Patch) Rendering

- We use bicubic as an example
- The simplest (naïve): convert curved patches into primitives that we always know how to render
- From curved surfaces to polygon quadrilaterals (non-planar) and/or triangles (planar)
- Surface evaluation at grid points
- This is straightforward but inefficient, because it requires many times of evaluation of \( s(u,v) \)
- The total number is \( \frac{1}{\delta u} \frac{1}{\delta v} \times 3 \)
Surface Rendering

- Parametric grids ($[0,1] \times [0,1]$) as a set of rectangles
Surface Rendering

- Better approach: precomputation

\[ s(u, v) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} M \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \]

- \( M \) is constant throughout the entire patch. The followings are the same along isoparametric lines

\[
\begin{bmatrix}
    u^3 & u^2 & u & 1 \\
    v^3 & v^2 & v & 1
\end{bmatrix}
\]

- Use one dimensional array to compute and store (evaluation only once)
Surface Rendering

- How about many patches: the array is unchanged, its sampling rate is the same, this is more useful
- How about adaptive sampling based on curvature information!!!
- How to compute normal at any grid point (approximation)

\[
\begin{align*}
& s_u(u, v) \times s_v(u, v) \\
= & \left( s(u + \delta u, v) - s(u, v) \right) \times \left( s(u, v + \delta v) - s(u, v) \right)
\end{align*}
\]
Regular Surface

- Generated from a set of control points.
Curve Network
Coons Patch

\[ s(0, v), s(1, v) \]
\[ s(u, 0), s(u, 1) \]
Coons Patch

$s(0, v), s(1, v)$
Coons Patch

\[ s(u,0), s(u,1) \]
Coons Patch

\[ s(0,v), s(1,v), s(u,0), s(u,1) \]
Coons Patch

• Bilinearly blended Coons patch

\[ (P)f = (P_1 \oplus P_2)f = (P_1 + P_2 - P_1 P_2)f \]
\[ (P_1)f = f(0, \nu)L_0^1(u) + f(1, \nu)L_1^1(u) \]
\[ (P_2)f = f(u, 0)L_0^1(v) + f(u, 1)L_1^1(v) \]

• Bicubically blended Coons patch

\[ (P_1)f = f(0, \nu)H_0^3(u) + f_u(0, \nu)H_1^3(u) + f_u(1, \nu)H_2^3(u) + f(1, \nu)H_3^3(u) \]
\[ (P_2)f = f(u, 0)H_0^3(v) + f_v(u, 0)H_1^3(v) + f_v(u, 1)H_2^3(v) + f(u, 1)H_3^3(v) \]
Coons Patch

\[ s(0,v), s_u(0,v) \]
\[ s(1,v), s_u(1,v) \]
\[ s(u,0), s_v(u,0) \]
\[ s(u,1), s_v(u,1) \]
Gordon Surfaces

- Generalization of Coons techniques
- A set of curves
  \[ f(u_i, v), \ i = 0, \ldots, \ n \]
  \[ f(u, v_j), \ j = 0, \ldots, \ m \]
- Boolean sum using Lagrange polynomials

\[
(P_1)f = \sum_{i=0}^{n} f(u_i, v)L_i^n(u)
\]
\[
(P_2)f = \sum_{j=0}^{m} f(u, v_j)L_j^m(v)
\]
\[
(P)f = (P_1 \oplus P_2)f = (P_1 + P_2 - P_1P_2)f
\]
Transfinite Methods

- Bilinearly blended Coons patch
  - Interpolate four boundary curves
- Bicubically blended Coons patch
  - Interpolate curves and their derivatives
- Gordon surfaces
  - Interpolate a curve-network
- Triangular extension
  - Interpolate over triangles
Triangular Surfaces
Recursive Subdivision Algorithm
Curve Mathematics (Cubic)

- **Bezier curve**

\[ c(u) = \sum_{i=0}^{3} p_i B_i^3(u) \]

- **Control points and basis functions**

\[
B_0^3(u) = (1 - u)^3 \\
B_1^3(u) = 3u(1 - u)^2 \\
B_2^3(u) = 3u^2(1 - u) \\
B_3^3(u) = u^3
\]

- **Image and properties of basis functions**
Recursive Evaluation

- Recursive linear interpolation

\[
\begin{align*}
(1 - u) & \quad (u) \\
p_0^0 & \quad p_1^0 & \quad p_2^0 & \quad p_3^0 \\
p_0^1 & \quad p_1^1 & \quad p_2^1 \\
p_0^2 & \quad p_1^2 \\
p_0^3 & = c(u)
\end{align*}
\]
Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation: \( c(0) = p_0, c(1) = p_n \)
- Binomial expansion theorem:
  \[
  ((1 - u) + u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1 - u)^{n-i}
  \]
- Convex hull: the curve is bounded by the convex hull defined by control points
Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation
  \[ c(0) = p_0, c(1) = p_n \]
- Binomial expansion theorem
  \[ ((1 - u) + u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1 - u)^{n-i} \]
- Convex hull: the curve is bounded by the convex hull defined by control points
Derivatives

- **Tangent vectors can easily evaluated at the end-points**
  \[ c'(0) = 3(p_1 - p_0); \ c'(1) = (p_3 - p_2) \]

- **Second derivatives at end-points can also be easily computed**:
  \[
  c^{(2)}(0) = 2 \times 3((p_2 - p_1) - (p_1 - p_0)) = 6(p_2 - 2p_1 + p_0)
  \]
  \[
  c^{(2)}(1) = 2 \times 3((p_3 - p_2) - (p_2 - p_1)) = 6(p_3 - 2p_2 + p_1)
  \]
Derivative Curve

• The derivative of a cubic Bezier curve is a quadratic Bezier curve

\[
c'(u) = -3(1-u)^2 p_0 + 3((1-u)^2 - 2u(1-u)) p_1 + 3(2u(1-u) - u^2) p_2 + 3u^2 p_3 = \\
3(p_1 - p_0)(1-u)^2 + 3(p_2 - p_1)2u(1-u) + 3(p_3 - p_2)u^2
\]
More Properties (Cubic)

- Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

\[
\begin{align*}
  c(\nu), & \quad \nu \in [0, u] \\
  c(\nu), & \quad \nu \in [u, 1] \\
  c_l(u), & \quad u \in [0, 1] \\
  c_r(u), & \quad u \in [0, 1]
\end{align*}
\]

- The control points for the left and the right are

\[
\begin{align*}
  & p_0, p_1, p_2, p_3 \\
  & p_0, p_0, p_0 \\
  & p_3, p_2, p_1, p_0
\end{align*}
\]
Barycentric Coordinates

\[ r + s + t = 1 \]

\[ V = rR + sS + tT \]

\[ tsr(S \rightarrow T); srt(T \rightarrow S); rts(S \rightarrow R) \]
Triangular Bezier Patch

• Triangular Bezier surface

\[ s(u, v) = \sum_{i+j+k=n} p_{i,j,k} B_{i,j,k}^{n}(r, s, t) \]

• Where \( r+s+t=1 \), and they are local barycentric coordinates

• Basis functions are Bernstein polynomials of degree \( n \)

\[ B_{i,j,k}^{n}(r, s, t) = \frac{n!}{i!j!k!} r^i s^j t^k \]
Triangular Bezier Patch

- **How many control points and basis functions:**
  \[
  \frac{1}{2} (n + 1)(n + 2)
  \]

- **Partition of unity**
  \[
  \sum_{i,j,k \geq 0} B_{i,j,k}^n(r,s,t) = 1
  \]

- **Positivity**
  \[
  B_{i,j,k}^n(r,s,t) \geq 0; r,s,t \in [0,1]
  \]
Recursive Evaluation

\[
\begin{align*}
    p_{i,j,k}^0 &= p_{i,j,k} \\
    p_{i,j,k}^l &= r p_{i+1,j,k}^{l-1} + s p_{i,j+1,k}^{l-1} + t p_{i,j,k+1}^{l-1} ; i + j + k = n - l, i, j, k \geq 0 \\
    s(u,v) &= p_{0,0,0}^n
\end{align*}
\]
## Properties

- Efficient algorithms
- Recursive evaluation
- Directional derivatives
- Degree elevation
- Subdivision
- Composite surfaces
Research Issues

- Continuity across adjacent patches
- Integral computation
- Triangular splines over regular triangulation
- Transform triangular splines to a set of piecewise triangular Bezier patches
- Interpolation/approximation using triangular splines
Triangular Bezier Surface
Recursive Evaluation
Control points (Cubic)

\[
\begin{align*}
    p_{0,3,0} \\
    p_{0,2,1} & \quad p_{1,2,0} \\
    p_{0,1,2} & \quad p_{1,1,1} & \quad p_{2,1,0} \\
    p_{0,0,3} & \quad p_{1,0,2} & \quad p_{2,0,1} & \quad p_{3,0,0}
\end{align*}
\]
Basis Functions (Cubic)

\[
\begin{array}{cccc}
sss \\
3sst & 3rss \\
3stt & 6rst & 3rrs \\
ttt & 3rtt & 3rrt & rrr
\end{array}
\]
Triangular Patch Subdivision
Triangular Domain
Triangular Coons-Gordon Surface

\[ r = 0; f(r, s, t) \]

\[ s = 0; f(r, 0, t) \]

\[ t = 0; f(r, s, 0) \]
Triangular Coons-Gordon Surface

\[ s = \text{const.} \]

\[ r = \text{const.} \]

\[ t = \text{const.} \]
Triangular Interpolation

\[(P_1) f = f(r, 0, t)L_0^1(\alpha) + f(r, s, 0)L_1^1(\alpha)\]
\[\alpha = \frac{s}{s + t}\]

\[(P_2) f = f(r, s, 0)L_0^1(\beta) + f(0, s, t)L_1^1(\beta)\]
\[\alpha = \frac{r}{r + t}\]

\[(P_3) f = f(0, s, t)L_0^1(\gamma) + f(r, 0, t)L_1^1(\gamma)\]
\[\alpha = \frac{r}{r + s}\]
Triangular Interpolation

• The Boolean sum of any two operators results the same!

\[
(P_{12}) f = (P_1 \oplus P_2) f \\
(P_{13}) f = (P_1 \oplus P_3) f \\
(P_{23}) f = (P_2 \oplus P_3) f
\]

• Use cubic blending functions for C1 interpolation!

\[
(Q_1)f = f(r,0,t)H_0^3(\alpha) + D_\alpha f(r,0,t)H_1^3(\alpha) + D_\alpha f(r,s,0)H_2^3(\alpha) + f(r,s,0)H_3^3(\alpha)
\]

\[
(Q_2)f = \ldots \ldots
\]

\[
(Q_3)f = \ldots \ldots
\]
Gregory’s Method

- Convex combination

\[
\begin{align*}
(T_1)f &= f(r, 0, t) + \alpha D_{\alpha} f(r, 0, t) \\
(T_2)f &= \ldots. \\
(T_3)f &= \ldots. \\
(T_{12})f &= (T_1 \oplus T_2)f \\
(T_{13})f &= (T_1 \oplus T_3)f \\
(T_{23})f &= (T_2 \oplus T_3)f \\
(T)f &= (a_1 T_{23} + a_2 T_{13} + a_3 T_{12})f \\
\end{align*}
\]

\[
\begin{align*}
a_1 &= \frac{s^2}{r^2 + s^2 + t^2} \\
a_2 &= \ldots. \\
a_3 &= \ldots. \\
\end{align*}
\]

- Generalize to pentagonal patch!
Triangular B-splines
Surface Properties

- Inherit from their curve generators
- More!
- Efficient algorithms
- Continuity across boundaries
- Interpolation and approximation tools
Spherical Parameterization
Spherical Parameterization
Possible Applications

- Shape classification
- Medical registration
- Smooth surface fitting
- Solving PDEs on surfaces
Shape Morphing
Morphing
Multiresolution Mapping

- Multiresolution morphing
Feature Mapping
Texture Mapping
Parametric Solids

- **Tricubic solid**
  \[ p(u, v, w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} a_{ijk} u^i v^j w^k \]
  \[ u, v, w \in [0,1] \]

- **Bezier solid**
  \[ p(u, v, w) = \sum_{i} \sum_{j} \sum_{k} p_{ijk} B_i(u) B_j(v) B_k(w) \]

- **B-spline solid**
  \[ p(u, v, w) = \sum_{i} \sum_{j} \sum_{k} p_{ijk} B_{i,I}(u) B_{j,J}(v) B_{k,K}(w) \]

- **NURBS solid**
  \[ p(u, v, w) = \frac{\sum_{i} \sum_{j} \sum_{k} p_{ijk} q_{ijk} B_{i,I}(u) B_{j,J}(v) B_{k,K}(w)}{\sum_{i} \sum_{j} \sum_{k} q_{ijk} B_{i,I}(u) B_{j,J}(v) B_{k,K}(w)} \]
Parametric Solids

• Tricubic Hermite solid
• In general
• Also known as “hyperpatch”
• Parametric solids represent both exterior and interior
• Examples
  – A rectangular solid, a trilinear solid
• Boundary elements
  – 8 corner points, 12 curved edges, and 6 curved faces

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

\[ u, v, w \in [0, 1] \]
Curves, Surfaces, and Solids

• Isoparametric curves for surfaces
  \[
  s(u, v), s(u_i, v), s(u, v_j) \\
  \text{if } u_i = \text{const.} \quad \text{and } v_j = \text{const.}
  \]

• Isoparametric curves for solids
  \[
  s(u, v, w), s(u_i, v_j, w), s(u_i, v, w_k), s(u, v_j, w_k)
  \]

• Isoparametric surfaces for solids
  \[
  s(u, v, w), s(u_i, v, w), s(u, v_j, w), s(u, v, w_k)
  \]
Curves, Surfaces, and Solids

- **Non-isoparametric curves for surfaces**

- **Non-isoparametric curves for solids**

- **Non-isoparametric surfaces for solids**

\[
\begin{align*}
\mathbf{s}(u(t), v(t), w(t)) &= [u(t) \quad v(t) \quad w(t)] \\
\mathbf{c}(t) &= \begin{bmatrix} u(t) \\ v(t) \end{bmatrix} \\
\mathbf{s}(u, v) &= \begin{bmatrix} u \\ v \end{bmatrix}
\end{align*}
\]

\[
\mathbf{s}(u, v, w) = \mathbf{s}(u(a,b), v(a,b), w(a,b))
\]
Surfaces of Revolution
Surfaces of Revolution

- Geometric construction
  - Specify a planar curve profile on y-z plane
  - Rotate this profile with respect to z-axis
- Procedure-based model
- What kinds of shape can we model?
- Review: three dimensional rotation w.r.t. z-axis

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
Surfaces of Revolution

- Mathematics: surfaces of revolution

\[
c(u) = \begin{bmatrix} 0 \\ y(u) \\ z(u) \end{bmatrix}
\]

\[
s(u, v) = \begin{bmatrix} -y(u) \sin(v) \\ y(u) \cos(v) \\ z(u) \end{bmatrix}
\]
Frenet Frames

- **Motivation:** attach a smoothly-varying coordinate system to any location of a curve
- **Three independent direction vectors for a 3D coordinate system:** (1) tangent; (2) bi-normal; (3) normal

\[
\begin{align*}
  t(u) &= \text{normalize} \ (c_u(u)) \\
  b(u) &= \text{normalize} \ (c_u(u) \times c_{uu}(u)) \\
  n(u) &= \text{normalize} \ (b(u) \times t(u))
\end{align*}
\]

- **Frenet coordinate system (frame) \((t,b,n)\) varies smoothly, as we move along the curve \(c(u)\)**
Frenet Coordinate System
Sweeping Surface
General Sweeping Surfaces

- Surface of revolution is a special case of a sweeping surface
- Idea: a profile curve and a trajectory curve
- Move a profile curve along a trajectory curve to generate a sweeping surface
- Question: how to orient the profile curve as it moves along the trajectory curve?
- Answer: various options

\[
c_1(u) \\
c_2(v)
\]
General Sweeping Surfaces

- Fixed orientation, simple translation of the coordinate system of the profile curve along the trajectory curve
- Rotation: if the trajectory curve is a circle
- Move using the “Frenet Frame” of the trajectory curve, smoothly varying orientation
- Example: surface of revolution
- Differential geometry fundamentals: Frenet frame
Frenet Swept Surfaces

- Orient the profile Curve (C1(u)) using the Frenet frame of C2(v)
  - Put C1(u) on the normal plane (n,b)
  - Place the original of C1(u) on C2(v)
  - Align the x-axis of C1(u) with \(-n\)
  - Align the y-axis of C1(u) with b

- Example: if C2(v) is a circle

- Variation (generalization)

- Scale C1(u) as it moves

- Morph C1(u) into C3(u) as it moves

- Use your own imagination!
Ruled Surfaces
Ruled Surfaces

- Move one straight line along a curve
- Example: plane, cone, cylinder
- Cylindrical surface
- Surface equation
  \[ s(u, v) = (1 - v)a(u) + vb(u) \]
  \[ s(u, v) = (1 - v)s(u,0) + vs(u,1) \]
  \[ s(u, v) = p(u) + vq(u) \]
- Isoparametric lines
- More examples
Developable Surfaces

- Deform a surface to planar shape without length/area changes
- Unroll a surface to a plane without stretching/distorting
- Example: cone, cylinder
- Developable surfaces vs. Ruled surfaces
- More examples???
Developable Surface
Summary

• Parametric curves and surfaces
• Polynomials and rational polynomials
• Free-form curves and surfaces
• Other commonly-used geometric primitives (e.g., sphere, ellipsoid, torus, superquadrics, blobby, etc.)
• Motivation:
  – Fewer degrees of freedom
  – More geometric coverage
Straight Line

\[ x + 2y - 4 = 0 \]
\[ x + 2y - 4 > 0 \]
\[ x + 2y - 4 < 0 \]
Straight Line

• Mathematics

\[ ax + by + c = 0 \]
\[ + \alpha (ax + by + c) = 0 \]
\[ - \alpha (ax + y + c) = 0 \]

• Example

\[ x + 2y - 4 = 0 \]
Circle

\[ x^2 + y^2 - 1 > 0 \]

\[ x^2 + y^2 - 1 < 0 \]

\[ x^2 + y^2 - 1 = 0 \]
Conic Sections

- Mathematics

- Examples
  - Ellipse
  - Hyperbola
  - Parabola
  - Empty set
  - Point
  - Pair of lines
  - Parallel lines
  - Repeated lines

\[ ax^2 + 2bxy + cy^2 + dx + ey + f = 0 \]

- Examples
  - Ellipse
    \[ 2x^2 + 3y^2 - 5 = 0 \]
  - Hyperbola
    \[ 2x^2 - 3y^2 - 5 = 0 \]
  - Parabola
    \[ 2x^2 + 3y = 0 \]
  - Empty set
    \[ 2x^2 + 3y^2 + 1 = 0 \]
  - Point
    \[ 2x^2 + 3y^2 = 0 \]
  - Pair of lines
    \[ 2x^2 - 3y^2 = 0 \]
  - Parallel lines
    \[ 2x^2 - 7 = 0 \]
  - Repeated lines
    \[ 2x^2 = 0 \]
Conics

- Parametric equations of conics
- Generalization to higher-degree curves
- How about non-planar (spatial) curves
Plane

\[ x + y + z - 1 = 0 \]
Plane and Intersection

\[ \mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c \]

\[ \mathbf{n} \]
Plane

- **Example**
  \[ x + y + z - 1 = 0 \]

- **General plane equation**
  \[ ax + by + cz + y = 0 \]

- **Normal of the plane**
  \[ \mathbf{n} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \]

- **Arbitrary point on the plane**
  \[ \mathbf{p}_a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \]
Plane

• **Plane equation derivation**

\[
(x - a_x)a + (y - a_y)b + (z - a_z)c = 0
\]

\[
a x + b y + c z - (a_x a + a_y b + a_z c) = 0
\]

• **Parametric representation** (given three points on the plane and they are non-collinear!)

\[
p(u, v) = p_a + (p_b - p_a)u + (p_c - p_a)v
\]
Plane

- **Explicit expression (if c is non-zero)**

\[
z = -\frac{1}{c} (ax + by + d)
\]

- **Line-Plane intersection**

\[
\begin{align*}
l(u) &= p_0 + (p_1 - p_0)u \\
(n)(p_0 + (p_1 - p_0)u) + d &= 0 \\
u &= -\frac{np_0}{np_1 - np_0} = -\frac{\text{plane }(p_0)}{\text{plane }(p_1) - \text{plane }(p_0)}
\end{align*}
\]
Circle

- Implicit equation: \( x^2 + y^2 - 1 = 0 \)
- Parametric function:
  \[
  \mathbf{c}(\theta) = \begin{bmatrix}
  \cos(\theta) \\
  \sin(\theta)
  \end{bmatrix}
  \]
  \[0 \leq \theta \leq 2\pi\]

- Parametric representation using rational polynomials (the first quadrant):
  \[
  x(u) = \frac{1 - u^2}{1 + u^2}
  \\
y(u) = \frac{2u}{1 + u^2}
  \]
  \[u \in [0, 1]\]

- Parametric representation is not unique!
The diagram illustrates a 3D function:

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

The equation \( f(x, y) = 0 \) represents the contour where \( z = 0 \), forming a surface in the 3D space.
Implicit Equations for Curves

- Describe an implicit relationship
- Planar curve (point set) \( \{(x, y) | f(x, y) = 0\} \)
- The implicit function is not unique

\[
\begin{align*}
\{(x, y) | +\alpha f(x, y) = 0\} \\
\{(x, y) | -\alpha f(x, y) = 0\}
\end{align*}
\]

- Comparison with parametric representation

\[
p(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix}
\]
Implicit Equations for Curves

• Implicit function is a level-set
  \[
  \begin{cases}
  z = f(x, y) \\
  z = 0
  \end{cases}
  \]

• Examples (straight line and conic sections)
  \[
  ax + by + c = 0 \\
  ax^2 + 2bxy + cy^2 + dx + ey + f = 0
  \]

• Other examples
  – Parabola, two parallel lines, ellipse, hyperbola, two intersection lines
Implicit Functions for Curves

- Parametric equations of conics
- Generalization to higher-degree curves
- How about non-planar (spatial) curves
Implicit Equations for Surfaces

- Surface mathematics
  \[ \{(x, y, z) \mid f(x, y, z) = 0\} \]
- Again, the implicit function for surfaces is not unique
  \[ \{(x, y, z) \mid +\alpha f(x, y, z) = 0\} \]
  \[ \{(x, y, z) \mid -\alpha f(x, y, z) = 0\} \]
- Comparison with parametric representation

\[
p(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}
\]
Implicit Equations for Surfaces

- Surface defined by implicit function is a level-set
  \[ \begin{align*}
  w &= f(x, y, z) \\
  w &= 0
  \end{align*} \]

- Examples
  - Plane, quadric surfaces, tori, superquadrics, blobby objects

- Parametric representation of quadric surfaces

- Generalization to higher-degree surfaces
Quadric Surfaces

- **Implicit functions**
- **Examples**
  - Sphere
  - Cylinder
  - Cone
  - Paraboloid
  - Ellipsoid
  - Hyperboloid
- **More**
  - Two parallel planes, two intersecting planes, single plane, line, point

\[ ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + jz + k = 0 \]

\[
\begin{align*}
  x^2 + y^2 + z^2 - 1 &= 0 \\
  x^2 + y^2 - 1 &= 0 \\
  x^2 + y^2 - z^2 &= 0 \\
  x^2 + y^2 + z &= 0 \\
  2x^2 + 3y^2 + 4z^2 - 5 &= 0 \\
  x^2 + y^2 - z^2 + 4 &= 0 
\end{align*}
\]

- **Sphere**
  \[ x^2 + y^2 + z^2 - r^2 = 0 \]
  \[
  x = r \cos(\alpha) \cos(\beta) \\
  y = r \cos(\alpha) \sin(\beta) \\
  z = r \sin(\alpha) \\
  \alpha \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]; \quad \beta \in [-\pi, \pi]
  \]

- **Ellipsoid**
  \[
  \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0 \\
  x = a \cos(\alpha) \cos(\beta) \\
  y = b \cos(\alpha) \sin(\beta) \\
  z = c \sin(\alpha) \\
  \alpha \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]; \quad \beta \in [-\pi, \pi]
  \]

- **Geometric meaning of these parameters**
Generalization

- Higher-degree polynomials

\[ \sum_i \sum_j \sum_k a_{ijk} x^i y^j z^k = 0 \]

- Non polynomials
Superquadrics

- Geometry (generalization of quadrics)
  \[
  \left(\frac{x}{a_1}\right)^{\frac{2}{s}} + \left(\frac{y}{a_2}\right)^{\frac{2}{s}} - 1 = 0
  \]

- Superellipse
  \[
  \left(\frac{x}{a_1}\right)^{\frac{2}{s_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{s_2}} + \left(\frac{z}{a_3}\right)^{s_1} - 1 = 0
  \]

- Superellipsoid

- Parametric representation

- What is the meaning of these control parameters?
Algebraic Function

- Parametric representation is popular, but...
- Formulation
  \[ \sum_i \sum_j \sum_k a_{ijk} x^i y^j z^k = 0 \]
- Properties...
  - Powerful, but lack of modeling tools
Algebraic Patch

Tetrahedron

Control point, weight

Algebraic patch
Algebraic Patch

- A tetrahedron with non-planar vertices

\[ \mathbf{V}_{n000}, \mathbf{V}_{0n00}, \mathbf{V}_{00n0}, \mathbf{V}_{000n} \]

- Trivariate barycentric coordinate \((r,s,t,u)\) for \(p\)

\[
\mathbf{p} = r \mathbf{V}_{n000} + s \mathbf{V}_{0n00} + t \mathbf{V}_{00n0} + u \mathbf{V}_{000n}
\]

\[ r + s + t + u = 1 \]

- A regular lattice of control points and weights

\[
\mathbf{p}_{ijkl} = \frac{i \mathbf{V}_{n000} + j \mathbf{V}_{0n00} + k \mathbf{V}_{00n0} + l \mathbf{V}_{000n}}{n}
\]

\[ i, j, k, l \geq 0; i + j + k + l = n \]
Algebraic Patch

- There are \((n+1)(n+2)(n+3)/6\) control points. A weight \(w(I,j,k,l)\) is also assigned to each control point.

- Algebraic patch formulation

\[
\sum_{i} \sum_{j} \sum_{k} \sum_{l=n-i-j-k} w_{ijkl} \frac{n!}{i!j!k!l!} r^i s^j t^k u^l = 0
\]

- Properties
  - Meaningful control, local control, boundary interpolation, gradient control, self-intersection avoidance, continuity condition across the boundaries, subdivision
Spatial Curves

• Intersection of two surfaces

\[ \begin{align*}
    f(x, y, z) &= 0 \\
    g(x, y, z) &= 0
\end{align*} \]
Algebraic Solid

- **Half space**
  \[
  \{(x, y, z) \mid f(x, y, z) \leq 0\}; \text{ or } \\
  \{(x, y, z) \mid f(x, y, z) \geq 0\}
  \]

- **Useful for complex objects (refer to notes on solid modeling)**

\[
f(x,y,z) = \begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \\ \Lambda \end{bmatrix} = \mathbf{0}
\]
Volume Datasets
Isosurface Rendering

Isovalue = 30
Isovalue = 100
Isovalue = 200
Direct Volume Rendering
Implicit Functions

- Long history: classical algebraic geometry
- Implicit and parametric forms
  - Advantages
  - Disadvantages
- Curves, surfaces, solids in higher-dimension
- Intersection computation
- Point classification
- Larger than parameter-based modeling
- Unbounded geometry
- Object traversal
- Evaluation
Implicit Functions

- Efficient algorithms, toolkits, software
- Computer-based shape modeling and design
- Geometric degeneracy and anomaly
- Algebraic and geometric operations are often closed
- Mathematics: algebraic geometry
- Symbolic computation
- Deformation and transformation
- Shape editing, rendering, and control
Implicit Functions

- Conversion between parametric and implicit forms
- Implicitization vs. parameterization
- Strategy: integration of both techniques
- Approximation using parametric models
Free-Form Deformation

• Free-Form Deformation Example

Original Model

Solid Mesh

Deformed Mesh

Result
Free-Form Deformation

- Free-Form Deformation Example (Complex >> 49000 faces)

Original Model

Solid Mesh

Deformed (Results in both surface rendered and wireframe)
Free-Form Deformation

- Free-Form Deformation Example (Non-trivial topology)

Original Model

Solid Mesh with a hole

Deformed Mesh

Result (no change in central cylinder)
Free-Form Deformation

- Free-Form Deformation Example (Localized)
Shape Modeling

• Direct Modeling / Manipulation
Material Modeling

- Material Representation (Non-homogeneous)