Curves: Review

Curves in 2D and 3D

- Implicit vs. Explicit vs. Parametric curves
- Bézier curves, de Casteljau algorithm
- Continuity
- B-Splines
- Subdivision Curves
Curves & Surfaces as Explicit Functions

Curves:
\[ y = F(x) \]

Surfaces:
\[ z = F(x, y) \]

Examples:
Not a function in Cartesian coord.,
\[ y = \pm \sqrt{1 - x^2} \]

Not representable as a function:

Limitations of explicit functions:
• Cannot model every curve in 2D
• No true 3D curves possible
  – All curves confined to a plane
Curves & Surfaces as Implicit Functions

**Curves**

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

**Surfaces**

\[ F(x, y, z) = 0 \]

**Interpretation for curves:**
- Iso-lines (contours) in a terrain

**Property:**
- If \( F \) is continuous, implicit curves and surfaces are always closed or extend to infinity

---

Conversion:
- Explicit to implicit: trivial
- Implicit to explicit: hard
  - Solving for \( y \) involves root finding!

**Limitations of implicit curves:**
- Curves only in 2D
  - Every implicit function in 3D describes a surface!
- Often unintuitive
- Difficult to render (display)
- But: useful for many tasks, including modeling, medical imaging
Curves & Surfaces as Parametric Functions

**Concept:**
- Curve as function of artificial “time” parameter $t$

**2D curve:**
\[
\begin{pmatrix}
 x \\
 y
\end{pmatrix} = \begin{pmatrix}
 F_x(t) \\
 F_y(t)
\end{pmatrix} =: F(t); F : \mathbb{R} \alpha \mathbb{R}^2
\]

**3D curve:**
\[
\begin{pmatrix}
 x \\
 y \\
 z
\end{pmatrix} = \begin{pmatrix}
 F_x(t) \\
 F_y(t) \\
 F_z(t)
\end{pmatrix} =: F(t); F : \mathbb{R} \alpha \mathbb{R}^3
\]

---

Curves & Surfaces as Parametric Functions

**Curve example:**
\[
\begin{pmatrix}
 x \\
 y \\
 z
\end{pmatrix} = \begin{pmatrix}
 \cos t \\
 \sin t \\
 t
\end{pmatrix}
\]

**Surfaces (in 3D):**
\[
\begin{pmatrix}
 x \\
 y \\
 z
\end{pmatrix} = \begin{pmatrix}
 F_x(s,t) \\
 F_y(s,t) \\
 F_z(s,t)
\end{pmatrix} = F(s,t); F : \mathbb{R}^2 \alpha \mathbb{R}^3
\]
Curves & Surfaces as Parametric Functions

This works in arbitrary dimensions!

• Curves:
  \[ \mathbf{x} = F(t); F : \mathbb{R} \rightarrow \mathbb{R}^d \]

• Surfaces:
  \[ \mathbf{x} = F(s, t); F : \mathbb{R}^2 \rightarrow \mathbb{R}^d \]

• Hypersurfaces:
  \[ \mathbf{x} = F(t); F : \mathbb{R}^n \rightarrow \mathbb{R}^d ; n < d \]

Notation:

• Bold variables (t, x) denote vectors, while italics denote scalars (t, d).

Splines: parametric curves over geometric base

Geometric meaning of coefficients (base)

• Approximate/interpolate set of positions, derivatives, etc.
Splines

Description = basis functions + coefficients

\[ F(t) = \sum_{i=0}^{n} P_i B_i(t) = (x(t), y(t)) \]

\[ x(t) = \sum_{i=0}^{n} P_i^{x} B_i(t) \]

\[ y(t) = \sum_{i=0}^{n} P_i^{y} B_i(t) \]

- Same basis functions for all coordinates

Parametric Spline Curves

Commonly used classes:

- Polynomials
  - Lagrange, Bézier, Hermite
- Piecewise polynomials
  - B-splines
- Rational and piecewise-rational curves
  - Rational Bézier curves, rational B-splines (NURBS)
Interpolate “Control” Points: Lagrange Polynomials

Use points we want to interpolate as basis

- Polynomial degree = number of input points

Basis Functions: Lagrange Polynomials

- Given: m+1 parameter values \( t_0 \ldots t_m \)
- Define

\[
L_i^m(t) := \prod_{j=0 \ldots m, j \neq i} \frac{t - t_j}{t_i - t_j}; i = 0 \ldots m
\]

\[
L_i^m(t_k) = \begin{cases} 
1; i = k \\
0; else 
\end{cases}
\]

- Lagrange spline

\[
F(t) = \sum_{i=0}^{m} L_i^m(t_j) \cdot b_i
\]
Other Option: Hermite Cubic Basis

Geometrically-oriented coefficients
• 2 positions + 2 tangents

Require $F(0)=P_0$, $F(1)=P_1$, $F'(0)=T_0$, $F'(1)=T_1$

Define basis function per requirement

$$F(t) = P_0 h_{00}(t) + P_1 h_{01}(t) + T_0 h_{10}(t) + T_1 h_{11}(t)$$

Hermite Cubic Basis

Can satisfy with cubic polynomials as basis

$$h_{ij}(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

Obtain - solve 4 linear equations in 4 unknowns for each basis function

$$h_{ij}(t)i, j = 0,1, t \in [0,1]$$

<table>
<thead>
<tr>
<th>curve</th>
<th>$F(0)$</th>
<th>$F(1)$</th>
<th>$F'(0)$</th>
<th>$F'(1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_{00}(t)$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_{01}(t)$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_{10}(t)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_{11}(t)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Hermite Cubic Basis

Four polynomials that satisfy the conditions

\[ h_{00}(t) = t^2(2t-3) + 1 \quad h_{01}(t) = -t^2(2t-3) \]
\[ h_{10}(t) = (t-1)^2 \quad h_{11}(t) = t^2(t-1) \]

Bézier Curves

Definition:
- Bézier curve is a polynomial curve that uses Bernstein polynomials as basis

\[ F(t) = \sum_{i=0}^{m} b_i B_i^m(t) \]
- \( b_i \) are called control points of Bézier curve
- Control polygon obtained by connecting control points with line segments
Bernstein Polynomials

\[ B_i^m(t) := \binom{m}{i} t^i (1 - t)^{m-i}; i = 0..m; t \in [0,1], \]

\[ \binom{m}{i} = \frac{m!}{(m-i)!i!} \]

Properties of Bézier Curves

- Endpoints \( b_0 \) and \( b_m \) of control polygon interpolated & corresponding parameter values are \( t=0 \) and \( t=1 \)
- Bézier curve is tangential to control polygon at endpoints
- Curve lies within convex hull of control points
- Curve is affine invariant
- There is a fast, recursive evaluation algorithm – de Casteljau algorithm
De Casteljau Algorithm

**Graphical Interpretation:**
- Determine point $F(1/2)$ for the cubic Bézier curve given by the following four points:

$$b_0^0, b_1^0, b_2^0, b_3^0$$

$F(1/2) = b_0^0 = F(1/2)$

**Observation**

*De Casteljau generates 2 new control polygons!*
- For parameter interval $[0,1/2]$, and $[1/2,1]$
- Can be used to recursively subdivide control polygon
Example

Cubic case:

Degree Elevation

Replace degree $m$ polynomial ($m+1$ control points) with degree $m+1$ polynomial ($m+2$ control points):

- New control points $b'_i$:

$$b'_i = \frac{1}{m+1} \left[ i \cdot b_{i-1} + (m+1-i)b_i \right]$$
Degree Elevation

Examples:

$m=2$

$m=3$

Derivatives of Bézier Curves

**Theorem (proof on board):**

- The derivative of a Bézier curve $F(t)$ is given as

$$F(t) := \sum_{i=0}^{m} B_i^m(t) \cdot b_i$$

is given as

$$F'(t) := m \cdot \sum_{i=0}^{m-1} B_i^{m-1}(t) \cdot (b_{i+1} - b_i)$$
Continuity

Def:
- A curve $F(t)$ is called $C^k$-continuous if its $k^{th}$ derivative $F^{(k)}(t)$ exists (i.e. is continuous) everywhere.

Note:
- Polynomial curves are infinitely continuous.

Def:
- Two curve segments $F(t)$ and $G(t)$ are called $C^k$-continuous at $t_0$ if their first $k$ derivatives match at $t_0$.

Bezier Continuity

- $C^0$: share end control points $b_m = b'_0$
- $C^1$: $b_m - b_{m-1} = b'_1 - b'_0$
- $G^1$: $b_m - b_{m-1}$ collinear to $b'_1 - b'_0$
**B-Splines**

Idea: Generate basis where functions are continuous cross domains

Control point controls set of basis functions (to preserve continuity)

Alternative view: continuous basis functions defined on several domains

---

**Uniform Cubic B-Spline Curves**

**Definition**

\[ C(t) = \sum_{i=0}^{n-1} P_i N_i^3(t) \quad t \in [0, n] \]

\[ N_i^3(t) = \begin{cases} 
  r^3 / 6 & r = t - i \quad t \in [i, i+1] \\
  (-3r^3 + 3r^2 + 3r + 1) / 6 & r = t - i - 1 \quad t \in [i+1, i+2] \\
  (3r^3 - 6r^2 + 4) / 6 & r = t - i - 2 \quad t \in [i+2, i+3] \\
  (1-r)^3 / 6 & r = t - i - 3 \quad t \in [i+3, i+4] 
\end{cases} \]
Uniform Cubic B-Spline Curves

For any $t \in [3,n]$
\[
\sum_{i=j-3}^{j} N_i^3(t) = 1
\]

For any $t \in [j,j+1]$ only 4 basis functions are non zero
\[
\sum_{i=0}^{n-1} N_i^3(t) = 1
\]

Any point on cubic B-Spline is affine combination of at most 4 control points

Boundary Conditions for B-Splines

B-Splines do not interpolate any control points
- in particular end points
B-Splines

Direct recursion formula:

\[ N_i^0(t) = \begin{cases} 
1 & ; u_i \leq t < u_{i+1} \\
0 & ; \text{else} 
\end{cases} \]

\[ N_i^l(t) = \frac{t - u_i}{u_{i+l} - u_i} \cdot N_i^{l-1}(t) + \frac{u_{i+l+1} - t}{u_{i+l+1} - u_{i+1}} \cdot N_{i+1}^{l-1}(t) \]

Note:
- Not an affine combination

NURBs

- B-Spline (B-spline basis)
- Non-Uniform – different interval lengths (knots)
- Rational – rational basis functions

\[ C(t) = \frac{\sum_{i=0}^{n-1} w_i P_i N_i^3(t)}{\sum_{i=0}^{n-1} w_i N_i^3(t)} \quad t \in [3, n] \]
Subdivision: Corner Cutting

Cubic B-Spline (corner cutting)
The 4-point scheme

Proving scheme works:

- Convergence
  - *Will do on board & more details later*
- Degree of continuity
- Affine invariance
  - *As long as weights sum to 1*
  - *Proof on board*
Subdivision Matrix

Example: Chaikin subdivision

\[
\begin{pmatrix}
P^i_0 \\
P^i_1 \\
P^i_2 \\
P^i_3 \\
\end{pmatrix}
= \begin{pmatrix}
1/4 & 3/4 & 0 & 0 \\
0 & 3/4 & 1/4 & 0 \\
0 & 1/4 & 3/4 & 0 \\
0 & 0 & 3/4 & 1/4 \\
\end{pmatrix}
\begin{pmatrix}
P^0_0 \\
P^0_1 \\
P^0_2 \\
P^0_3 \\
\end{pmatrix}
\]