Curves and Surfaces

Chapter 9

Surface and Curve Representation

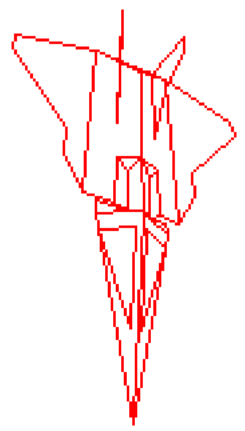
- Modeling existing objects

  Mathematical descriptions may be unavailable

- Creating models from scratch

  Perhaps from empirical data from scientific experiments

Polygon Mesh



- A set of connected planar surfaces
- Solids are bounded by polygon meshes
- Curved surfaces are only approximated by polygon meshes

Parametric Equations

Parametric Polynomial Curves

- Three polynomials with parameter *t*
- Coefficients of the polynomial determine the path of the curve
- Cubic polynomials are ideal
- Generally referred to as *cubic curves*

Polynomial Surface Patches

- Define the coordinates of points on a curved 3D surface
- Bivariate polynomials are used
- Generally referred to as bicubic patches

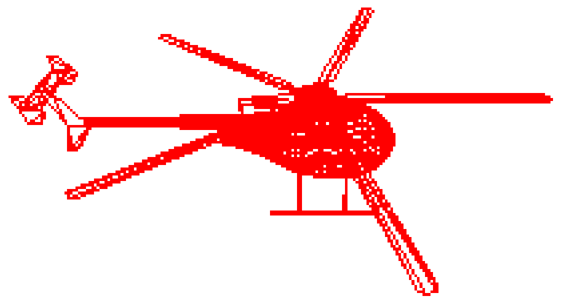
Quadric Surfaces

Useful for representing regular shapes

- Spheres
- Ellipsoids
- Cylinders

Polygon Meshes

- Collection of edges, vertices, are polygons
- Each edge is shared by at most two polygons
- An edge connects two vertices
- Every edge is part of some polygon



Representing Polygon Meshes

- Polygon meshes can be represented several ways
- There are advantages and disadvantages of each of representation
- Time vs. space trade-offs
- Application determines the best representation

Typical Polygon Operations

- Finding all edges incident to a vertex
- Finding the polygons sharing an edge
- Finding the polygons sharing a vertex
- Finding the vertices connected by an edge
- Displaying the mesh
- Checking the mesh’s consistency

Explicit Representation

- Each polygon is represented by a list of vertex coordinates

$$\{(x_0, y_0, z_0), (x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$$

- Vertices are stored in traversal order
- Edges exit between successive vertices in the list
- Good for simple polygons
  - Space efficient
- Bad for polygon meshes
  - Shared vertices are duplicated
  - Shared edges and vertices are not explicitly represented

Explicit Representation (cont.)

What happens if we attempt to drag a vertex interactively?

- We must find all polygons that share this vertex
- Therefore we must compare the vertices of one polygon with those of all the other polygons
- $n\ log\ n$ at best, and other problems make this impractical

Mesh Vertex List

- Each vertex is stored only once in a list
- A polygon is defined by a list of indices (pointers) into the vertex list
- Advantages
  - Vertices not duplicated
  - Vertex coordinates can be easily changed
- Disadvantages
  - Hard to find polygons that share an edge
  - Shared edges are drawn twice

Mesh Edge List

- Has a vertex list (like vertex list representation)
- Also has an edge list
- Edge list stores pairs of vertices the define that edge
- Edge list also stores up to two polygons that use that edge
- Redundant clipping, transformation, and scan conversion are avoided
- Filled polygons are easily displayed

Parametric Cubic Curves

- Express curve as a function of x, y, and z:

$$\begin{aligned} x &= x \\ y &= f(x) \\ z &= g(x) \end{aligned}$$

- Problem
  - Infinite slope at some points of the curve
    (infinity is tough to represent)
  - Plotting the curve smoothly requires that the slope be computed

Parametric Equations

As a function of t the slopes are replaced with tangent vectors that need never be infinite

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z \end{aligned}$$

$$0 \leq t \leq 1$$

Tangent Vectors

Shown for $x(t)$, but works similarly for $y(t)$, $z(t)$

Find derivative with respect to t

$$\frac{dx}{dt} = 3a_x t^2 + 2b_x t + c_x$$

Three derivatives form the tangent vector

Slopes of the curves are ratios of the tangent vectors

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt}$$

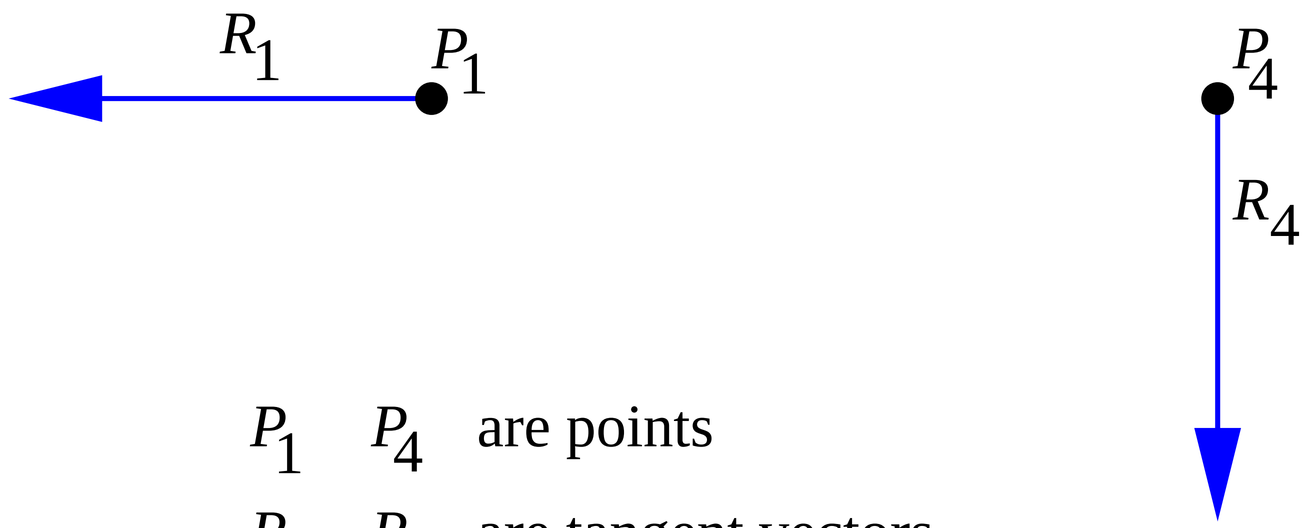
$$\frac{dx}{dz} = \frac{dx/dt}{dz/dt}$$

etc . . .

Why Cubic Curves?

- No lower order representation of curve segments can provide continuity of position and slope where curve segments meet and at the same time ensure the ends of the curve segment pass through the specified points
- Lowest order representation that can describe a non-planar curve (required for 3D curves)
- Higher order parametric curves produce undesirable wriggles

Hermite Form of a Cubic



P_1 P_4 are points

R_1 R_4 are tangent vectors

Hermite Math (1)

$$\begin{aligned} x(0) &= P_{1_x} \\ x(1) &= P_{4_x} \\ x'(0) &= R_{1_x} \\ x'(1) &= R_{4_x} \end{aligned}$$

Subscript refers to x coordinate of point or tangent vector

Need to find a_x, b_x, c_x, and d_x

Hermite Math (2)

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} C_x = T C_x$$

C_x is the column vector of coefficients of $x(t)$

T is the row vector of powers of t

Hermite Math (3)

Conditions (1) expressed by (2):

$$x(0) = P_{1_x} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} C_x$$

$$x(1) = P_{4_x} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} C_x$$

Hermite Math (4)

To get the tangent vectors, differentiate (2)

$$x'(t) = [\ 3t^2 \quad 2t \quad 1 \quad 0\ ]C_x$$

$$x'(0) = R_{1_x} = [\ 0 \quad 0 \quad 1 \quad 0\ ]C_x$$

$$x'(1) = R_{4_x} = [\ 3 \quad 2 \quad 1 \quad 0\ ]C_x$$

Hermite Math (5)

Combine (3) and (4) to get

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} C_x$$

Hermite Math (6)

Invert the 4×4 matrix from (5) and multiply it to both sides of (5)

$$C_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = M_h G_{h_x}$$

M_h is called the *Hermite matrix*

G_h is called the *Hermite geometry matrix*

Hermite Math (7)

$$\begin{aligned} x(t) &= TM_hG_{h_x} \\ y(t) &= TM_hG_{h_y} \\ z(t) &= TM_hG_{h_z} \end{aligned}$$

Or, grouped together as

$$P(t) = TM_hG_h$$

Given P_1, P_4, R_1, and R_4, we can evaluate $x(t)$, $y(t)$, and $z(t)$ for $0 \leq t \leq 1$ and find all the points on the cubic curve.

$$TM_h = [\ (2t^3 - 3t^2 + 1) \quad (-2t^3 + 3t^2) \quad (t^3 - 2t^2 + t) \quad (t^3 - t^2)\ ]$$

Hermite Math (8)

$$TM_h = [\ (2t^3 - 3t^2 + 1) \quad (-2t^3 + 3t^2) \quad (t^3 - 2t^2 + t) \quad (t^3 - t^2)\ ]$$

$$\begin{aligned} x(t) &= TM_h G_{h_x} \\ &= P_{1_x}(2t^3 - 3t^2 + 1) + P_{4_x}(-2t^3 + 3t^2) + R_{1_x}(t^3 - 2t^2 + t) \\ &\quad + R_{4_x}(t^3 - t^2) \end{aligned}$$

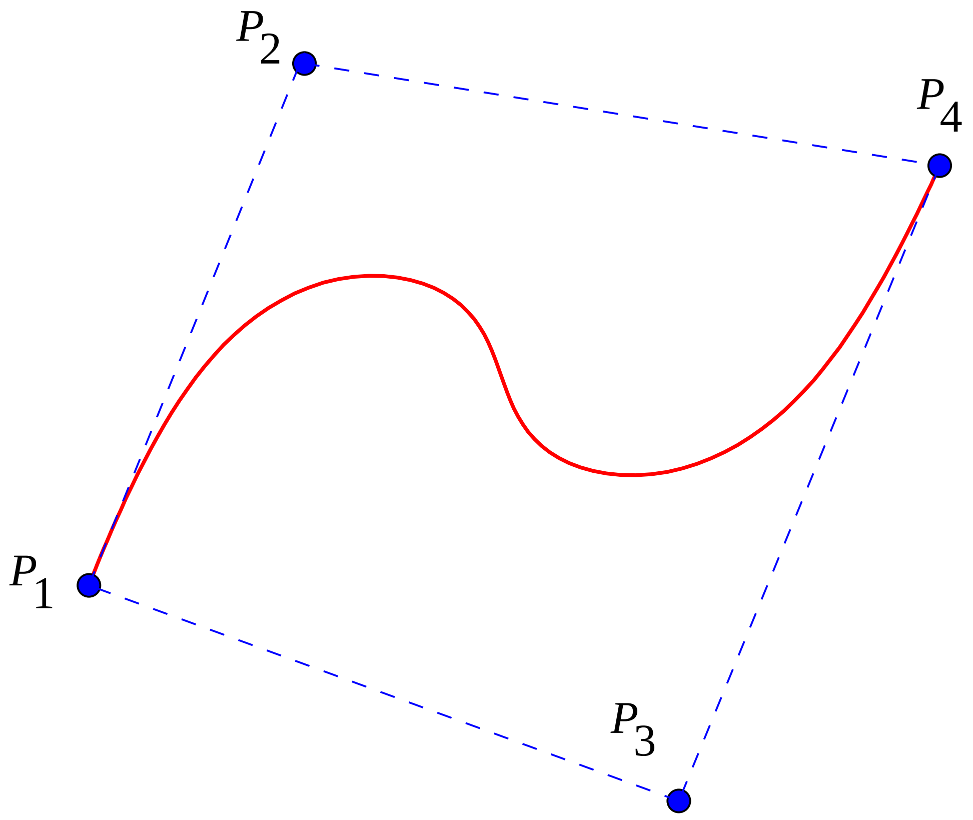
The four terms are called four “blending functions”

The first two blend P_1 and P_4

The second two blend R_1 and R_4

Bezier Form

- Similar to Hermite form but differs in definition of tangent vectors
- Four points are used



Difference from Hermite

Tangent vectors of the endpoints e determined by the lines $\overline{P_1P_2}$ and $\overline{P_3P_4}$.

R_1 and R_4 of the Hermite form are related to the Bezier points P_1, P_2, P_3, and P_4 by:

$$\begin{aligned} R_1 &= 3(P_2 - P_1) &= P'(0) \\ R_4 &= 3(P_4 - P_3) &= P'(1) \end{aligned}$$

Bezier Geometry Matrix

$$G_h = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}_x = M_{hb} G_b$$

Bezier Matrix

Recall $x(t) = TM_hG_{h_x}$

$$x(t) = TM_hG_{h_x} = TM_hM_{hb}G_{b_x}$$

Let $M_hM_{hb} = M_b$.

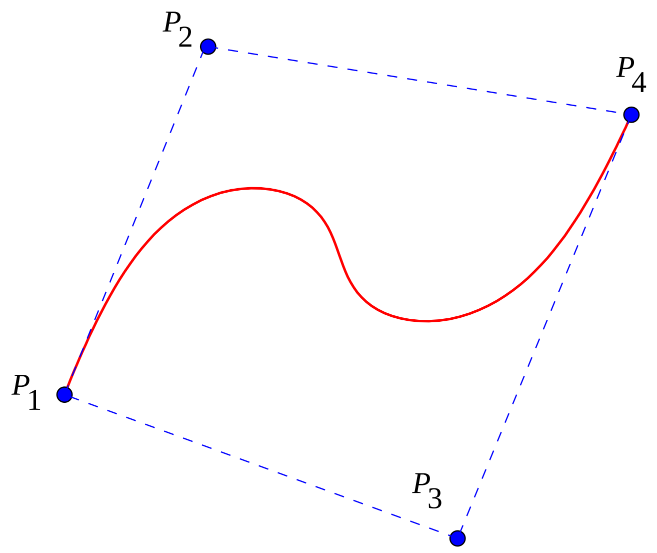
$$x(t) = TM_bG_{b_x}$$

$$M_b = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Why use Bezier Form?

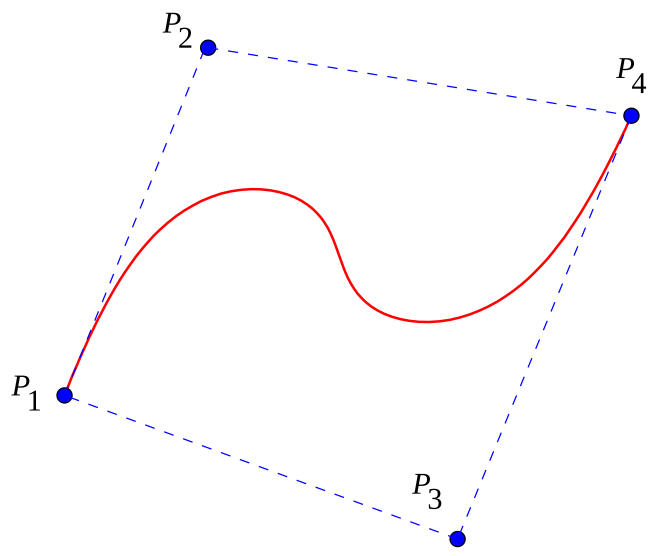
Why use Bezier form instead of Hermite form?

- Four control points for interactive use (moving with a mouse, for instance)
- Four control points bound the curve (convex hull)
- Can be used to clip the curve



B-spline Form

- “Smoother” than the other forms
- Concept similer to draftsman’s splines



B-spline Matrix

B-spline forms:

$$x(t) = T M_s G_{s_x}$$

where

$$M_s = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

B-spline Matrix

Approximate control points $P_1, P_2, \ldots, P_n$ by a series of B-splines.

Use a *different* geometry matrix between each pair of adjacent points.

$$G_s^i = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}, \quad 2 \leq i \leq n-2$$

Interpolation

- Presented as a lead in to B-splines
- Use an array of sample points to approximate a curve
- Could use polynomial, trigonometric, or exponential functions to build a curve
- We’ll use polynomial functions

Interpolation (1)

n points $(P_1, P_2, \ldots, P_n)$, which is shorthand for $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$

$$x(t) = \sum_{i=1}^{n} x_i b_i(t)$$

$$y(t) = \sum_{i=1}^{n} y_i b_i(t)$$

$$z(t) = \sum_{i=1}^{n} z_i b_i(t)$$

$b_i(t)$ are blending functions

Blending Functions

$$b_i(t)$$

For each value of t they determine how much the ith sample point affects the position of the curve.

Think of each sample point as trying to pull the curve in its direction.

$b_i(t)$ tells how hard the ith sample point is pulling.

If for some t, $b_i(t) = 1$ and for each $j \neq i, b_j(t) = 0$, then the ith sample point has complete control over the curve. The curve therefore *will* pass through point i.

Blending Functions (cont.)

For convenient formulation:

Let P_1 be in complete control when $t = -1$

Let P_2 be in complete control when $t = 0$

Let P_3 be in complete control when $t = 1$

$\ldots$

$b_1(t) = 1$ at $t = -1$

$b_1(t) = 0$ at $t = 0, 1, \ldots, n-2$

Blending Functions (cont.)

Consider the function:

$$t(t-1)(t-2)\dots(t-[n-2])$$

At $t=-1$, it is

$$(-1)(-2)(-3)\dots(1-n)=c$$

Dividing by c will yield 1

$$b_1(t)=\frac{t(t-1)(t-2)\dots(t-[n-2])}{(-1)(-2)(-3)\dots(1-n)}$$

Blending Functions (cont.)

In general, for i:

$$b_i(t) = \frac{(t+1)(t)(t-1)\dots(t-[i-3])(t-[i-1])\dots(t-[n-2])}{(i-1)(i-2)(i-3)\dots(1)(-1)(i-n)}$$

Which is 1 at $t = i-2$ and 0 for all other integers

n points $(P_1, P_2, \dots, P_n)$i, which is shorthand for $(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$

Blending Functions (cont.)

Need four blending functions for four points:

$$
\begin{aligned}
b_1(t) &= \frac{t(t-1)(t-2)}{(-1)(-2)(-3)} \\
b_2(t) &= \frac{(t+1)(t-1)(t-2)}{(1)(-1)(-2)} \\
b_3(t) &= \frac{(t+1)t(t-2)}{(2)(1)(-1)} \\
b_4(t) &= \frac{(t+1)t(t-1)}{(3)(2)(1)}
\end{aligned}
$$

Blending Functions (cont.)

$$\begin{aligned}
x(t) &= x_1 b_1(t) + x_2 b_2(t) + x_3 b_3(t) + x_4 b_4(t) \\
y(t) &= y_1 b_1(t) + y_2 b_2(t) + y_3 b_3(t) + y_4 b_4(t) \\
z(t) &= z_1 b_1(t) + z_2 b_2(t) + z_3 b_3(t) + z_4 b_4(t)
\end{aligned}$$

The curve should lie close to the actual curve in the region of the four points, especially in the middle between P_2 and P_3.

This is where $0 \le t \le 1$.

Shortcomings

The sum of the blending functions is not 1 at all values of t.

We've designed them to sum to 1 at *integer* values of t, not in between.

Suppose all sample points had the same x value: $x = x_0$.

We would expect the curve to have constant values for x at points in between the sample points.

Shortcomings (cont.)

However, approximating the curve yields:

$$x(t) = \sum_{i=1}^{n} x_i b_i(t)$$

If $x_i = x_0$:

$$x(t) = x_0 \sum_{i=1}^{n} b_i(t)$$

The line may wiggle in and out of the plane.

Shortcomings (cont.)

- Each section of the curve is connected to the next section through a point and the curves may have different slopes at the two sections
- Transition of control is not smooth at point P_i is approached, the control of the other points is reduced to zero

A Better Way

- A more natural approach to have a sample point control the curve is to have the blending function vary from 0 far away from the point to some maximum value (not necessarily 1) near the point
- We don’t force the curve through a particular point but gently attract it towards the point
- The resulting curve will follow the general contours of the control points but may not actually pass through any of them
- The set of blending functions that achieve this and always sum to 1 are called *B-splines*