Representing Curves and Surfaces

CSC 292:
Computer Graphics

Randal C. Nelson

Needs

Want to represent smooth curves and surfaces

- Real world objects.
- CAD
- Plots, sketches.
- Animation, camera paths.
- Interpolation of datasets in modeling.
- Interactive creation of new models.

We will consider various techniques

- Polygon meshes.
- Parametric curves (cubic).
- Parametric surfaces (bicubic).
- Quadric surfaces.

Polygon Meshes

Approximate curved surfaces with *connected polygons*.

- Collection of edges, vertices, and polygons.
- Edges connect 2 vertices.
- Polygon is (planar) closed sequence of edges.
- Each edge is shared by at most 2 polygons.

Various operations must be supported, depending on application, but some are common.

- Find edges incident to vertex.
- Find all polygons sharing edge or vertex.
- Find vertices connected by an edge.
- Find edges of polygon.
- Display mesh in various ways.
- Find errors (e.g. missing edges).
- Compute max distances from polygons to other surface representation.

Representing Polygon Meshes

Various representations have been used, all with various tradeoffs.

- Explicit polygon set.
- Pointers to vertex list.
- Pointers to edge list.

Explicit Polygons

- Store polygons as *lists of vertices*.
- Simple.
- Edges and vertices are duplicated.
- No explicit representation of shared edges and vertices, so they are hard to find.
- Match problems due to roundoff.
- Shared edges drawn twice.

Pointers to Vertex List

- SPHIGS method.
- Vertices stored once.
- Polygons are *lists of pointers to vertices*.
- Easy sharing of vertices.
- Still hard to find polygons that share an edge.
- Edges still drawn twice.

Pointers to an edge list

- Set of vertices.
- Set of edges where each edge contains pointers to vertices and pointers to polygons.
- Polygons are *lists of pointers to edges*.
- Display by showing all edges.
- Can be extended to allow more than 2 polygons to intersect at an edge.
- Still doesn't allow easy determination of edges incident to a vertex.
- Modification called a *winged edge* extends edges by adding pointers to adjacent edges, and extends vertices to include pointers to incident edges.

Consistency Checking

Check that:

- Polygons are closed.
- Each edge is used at least once, but not more than (twice, n-times).
- Each vertex is referenced by at least two edges

Sometimes we want to ensure that:

- Mesh is *connected*.
- Graph is *planar*.
- Polyhedron has no holes.

Watch for:

- Zero length edges.
- Same edge twice in a polygon.
- Overall consistency in more complex representations.

Finding Plane Equations

Often we want to know the *equation* of the plane containing a polygon. Recall that the general equation for a plane is:

$$Ax + By + Cz + D = 0$$

- Three non-collinear points define a plane.
- So one way is to *pick three vertices*.
- A, B, and C define the plane normal and can be found with cross product of two vectors connecting vertices. This also checks collinearity.
- Obtain D by plugging in one point.

More Robust Method

Polygons with more than 3 points may not be planar due to numerical (roundoff) errors, or errors in generation process

- In almost planar case, can use all points to get better guess.
- Use fact that A, B, and C are proportional to signed areas of projection of polygon onto plane perpendicular to corresponding axis.
- Projection is easy.
- Polygon areas are computed from (signed) sums of trapezoids.
- Compute non-planarity by finding perpendicular distances.

$$d = \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}}$$

- Representation is not unique.

Parametric Cubic Curves

- Using *linear approximations* (segments, polygons) takes lots of points to get good accuracy for curved objects.
- Can be a problem for some applications (especially interactive generation).
- Using *higher degree polynomials* can give better (and better-looking) approximations with much less explicit information.

Representations for Equations

- Explicit representation: $y = f(x)$
  - Can't do multiple values of y - have to break into pieces.
  - These pieces are not rotationally invariant.
  - Vertical tangents cause problems.
- Implicit equations: $x^2 + y^2 = 1$
  - Might have more solutions that we want.
  - Solving is tricky.
  - Joining is tricky.
- Parametric equations: $x = f(t); y = g(t)$
  - Avoids above problems.
  - Typically what is used.

Parametric Cubic Curves

- We will use *piecewise polynomial* curves, each segment represented parametrically, with constraints on joins.
- Use *cubics* because they are the lowest degree with enough flexibility to match endpoints and slope.
- Higher orders give unwanted wiggles and are harder to use.
- 4 parameters of cubic let 4 geometric parameters be set (e.g. 2 endpoints and 2 slopes).
- Parametric cubics can be non-planar in 3D (2nd order curves always lie in some plane).
- Sometimes higher degree polynomials are used in applications where more constraints must be met (aerodynamics)

Mathematical Representation

- Parametric cubic polynomials have the form

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

  with t on the interval $[0, 1]$.

- If we define the vector $T = [t^3, t^2, t, 1]$ and the matrix

$$
C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}
$$

  Then we can write concisely
  $Q(t) = [x(t), y(t), z(t)] = T \cdot C$

Continuity Properties

- The *derivative* of the curve is the tangent vector, and is given by $Q'(t) = T' \cdot C = [3t^2, 2t, 1, 0] \cdot C$.
- Tangent has both direction and magnitude.
- Two types of continuity are defined for parametric curves: *geometric* continuity G^n, and *parametric* continuity C^n.
- G^0 geometric continuity means the endpoints match.
- G^1 geometric continuity means the tangent directions (but not necessarily the magnitudes) match.

- C^n continuity means the nth derivatives match.
- C^0 means the endpoints match (just like G^0).
- C^1 means the tangent vectors direction *and* magnitude match. i.e. no discontinuities in *velocity* if t is time.
- C^2 means no *acceleration* discontinuities.

Continuity Constraints

- Visually, when Q generates a static curve, G^1 is enough, and indistinguishable from C^1.
- For *animation*, we generally want at least C^1 and sometimes C^2.
- Generally, C^1 implies G^1 (not vice versa). Exception if magnitude goes to 0.
- Curve segments are determined by *constraints* on endpoints, tangent vectors, and continuity.
- Each cubic polynomial has 4 unknowns, so we need 4 constraints per segment.

Cubic Fitting Methods

We will look at several different methods for approximating curves (and ultimately surfaces) using cubic polynomials, with different sorts of constraint mechanisms.

- *Hermite* - 2 endpoints, 2 tangents.
- *Bezier* - 2 endpoints, 2 tangent-controlling control points
- *Splines* (several types) controlled by 4 control points that lie near, but are not generally interpolated by the spline.

Generic Form

- We rewrite the coefficient matrix C as a product of two matrices $C = M \cdot G$.
- M is a 4×4 matrix called the *basis matrix*.
- G is a $4 \times n$ *geometry matrix* composed of a column of n-dimensional constraints (such as endpoints or tangent vectors).

$$C = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} g_{1x} & g_{1y} & g_{1z} \\ g_{2x} & g_{2y} & g_{2z} \\ g_{3x} & g_{3y} & g_{3z} \\ g_{4x} & g_{4y} & g_{4z} \end{bmatrix}$$

- If we multiply out just $x(t) = T \cdot M \cdot G_x$, we get a big long equation:

$$\begin{aligned} x(t) \;=\; & (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1x} + \\ & (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2x} + \\ & (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3x} + \\ & (t^3 m_{14} + t^2 m_{23} + t m_{34} + m_{44}) g_{4x} \end{aligned}$$

- $y(t)$ and $z(t)$ are analogous.
- We note the above is just a *weighted sum* of cubic equations. The equations are determined by M and the weights are the control points.
- The different cubic polynomials are called *blending functions* and are given by $B = T \cdot M$.

- Note similarity to the linear interpolation formula

$$
\begin{aligned}
x(t) &= (1-t)g_{1x} + (t)g_{2x} \\
y(t) &= (1-t)g_{1y} + (t)g_{2y} \\
z(t) &= (1-t)g_{1z} + (t)g_{2z}
\end{aligned}
$$

- In this sense, fitting cubics can be seen as a simple generalization of fitting line segments.

Representation of Fitting Methods

- The different fitting methods can all be represented by varying M and the interpretation of the points composing G.
- The elements of M are carefully determined to guarantee specific continuity conditions on the interpretation of G.
- Finding the M matrices is a bit of a trick, but it only has to be derived once for each fitting method.
- Generalizes to higher degree polynomials.

Hermite Curves

- Constraints are on endpoints P_1 and P_4, and tangent vectors (magnitude included) R_1 and R_4 at those same endpoints (indexing is for consistency with later models).
- Together these make up the *Hermite geometry vector* (actually a matrix) G_H.

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} p_{1x} & p_{1y} & p_{1z} \\ p_{4x} & p_{4y} & p_{4z} \\ r_{1x} & r_{1y} & r_{1z} \\ r_{4x} & r_{4y} & r_{4z} \end{bmatrix}$$

- We want to find the *Hermite basis matrix* M_H that relates G_H to the polynomial coefficients.
- This can be done by setting up constraints and solving for the elements of M_H.
- First recall that

$$x(t) = [\, t^3 \;\; t^2 \;\; t \;\; 1\, M_H \cdot G_{H_x}]$$

- By substituting $t = 0$ and $t = 1$ into the equations for $x(t)$ and $x'(t)$, we get four equations:

$$\begin{aligned}
x(0) &= P_{1_x} = [\,0 \;\; 0 \;\; 0 \;\; 1\,] M_H \cdot G_{H_x} \\
x(1) &= P_{4_x} = [\,1 \;\; 1 \;\; 1 \;\; 1\,] M_H \cdot G_{H_x} \\
x'(0) &= P_{1_x} = [\,0 \;\; 0 \;\; 1 \;\; 0\,] M_H \cdot G_{H_x} \\
x'(1) &= P_{1_x} = [\,3 \;\; 2 \;\; 1 \;\; 0\,] M_H \cdot G_{H_x}
\end{aligned}$$

- These can be rewritten in matrix form

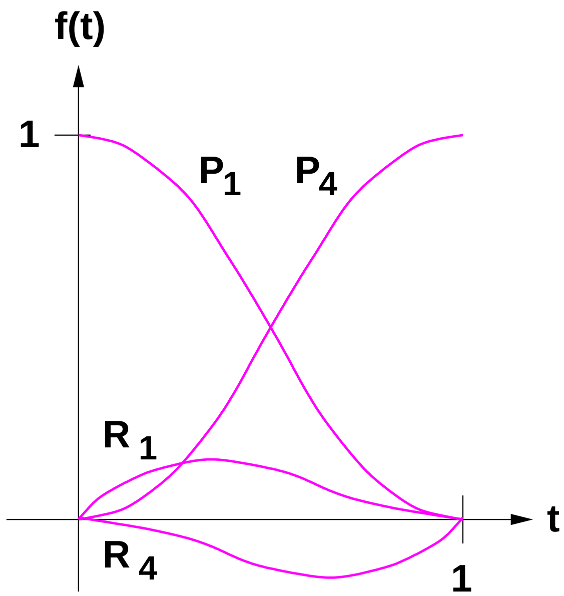
$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = G_{H_x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

- For this to be true M_H must be the inverse of the 4×4 matrix

$$M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- If we expand this, we get the *Hermite blending functions*:

$$\begin{aligned} Q(t) &= (2t^3 - 3t^2 + 1)P_1 \\ &+ (-2t^3 + 3t^2)P_4 \\ &+ (t^3 - 2t^2 + t)R_1 \\ &+ (t^3 - t^2)R_4 \end{aligned}$$



- Note that at $t = 0$ only P_1 effects the value, and at $t = 1$ only P_4 does. In between, the other polynomials contribute.

- By varying the tangent slope and magnitude, the character of the curve passing through the endpoints can be modified.
- For passing a curve through multiple points, we generate one Hermite cubic per interval and control the tangents to obtain the desired continuity (G^1 or C^1).
- A simple method of display is just to increment t and find values on the curve by direct computation (a bit inefficient).
- We can transform the curves by transforming the geometry vector and using it to generate the transformed curve.

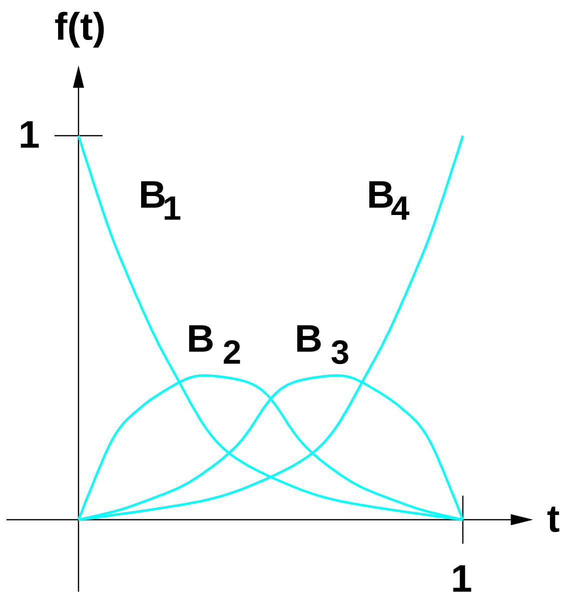
Bezier Curves

- Here, instead of two endpoints and two tangents, we specify two endpoints P_1 and P_4, and two control points P_2 and P_3 that indirectly specify the tangents.
- Specifically $R_1 = 3(P_2 - P_1)$ and $R_4 = 3(P_4 - P_3)$.
- The factor of 3 gives good continuity properties.
- Curve interpolates the endpoints, and approximates the control points.
- Running a similar derivation gives us the basis matrix M_B for the Bezier geometry matrix G_B.

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

- The corresponding blending functions are called the *Bernstein Polynomials*.

$$\begin{aligned} Q(t) &= (1-t)^3 P_1 \\ &+ 3t(1-t)^2 P_2 \\ &+ 3t^2(1-t) P_3 \\ &+ t^3 P_4 \end{aligned}$$



- Note symmetry

- Points are non-negative weighted average of control points.
- If weights always sum to one, as here, then whole curve lies inside the *convex hull* of the control points.
- Continuity at joins requires point and control points on either side to be collinear.

Uniform Nonrational B-Splines

- Originally, *splines* were strips of metal bent with weights, and used to lay out sufaces of cars, airplanes etc.
- Metal strips have second order continuity.
- The *natural cubic spline* is C^2 continuous polynomial that interpolates all the control points. Non-local, and solution involves inverting $n + 1 \times n + 1$ matrix.
- Simpler *B-splines* are curve segments whose coefficients depend on just a few control points.

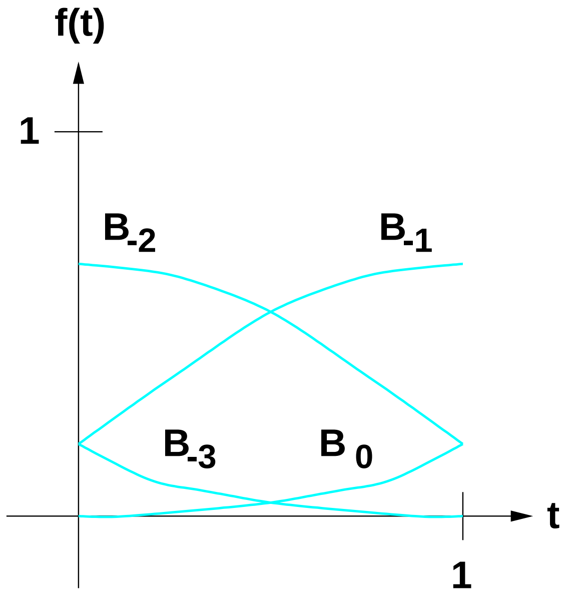
- Approximate a series of $m+1$ control points $P_0, P_1, ...P_m$ with $m-2$ cubic curve segments $Q_3, Q_4, ...Q_m$.
- Adjust parameters of each segment so that range of t is contiguous over whole curve.
- Each segment is thus associated with a range of t.
- Curve segments are joined at *knots*, and the parameter value at such a join is called the *knot value*. Initial and final points are also knots.
- The qualifier *uniform* means knots are spaced regularly in t. Thus without loss of generality, we can let the interval between knots be 1.

- Each segment is defined by 4 contiguous control points. Lots of overlap.
- Each control point affects 4 segments, so moving a control point moves the 4 segments in the same direction.
- By requiring C^0, C^1, and C^2 continuity, we can derive a basis matrix M_{Bs} and blending functions for the geometry vector G_{Bs} consisting of the 4 control points as before.

$$G_{Bs_i} = \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}; M_{Bs} = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

- The blending functions, now applied to contiguous groups of 4 points, are given by

$$
\begin{aligned}
Q(t-t_i) = & \frac{(1-t)^3}{6}P_{i-3} \\
+ & \frac{3t^3+6t^2+4}{6}P_{i-2} \\
+ & \frac{-3t^3+3t^2+3t+1}{6}P_{i-1} \\
+ & \frac{t^3}{6}P_i
\end{aligned}
$$



- Blending functions sum to one, so convex hull condition holds.
- Provides C^2 continuity, but at cost of less flexibility.