Modeling Curves

Curve representations

Two most common representations:

- **implicit e.g.** $x^2 + y^2 = 1$
  - hard to draw
  - easy to determine inside/outside
- **parametric, e.g.** $x = \cos t,\ \ y = \sin t$
  - easy to draw
  - many different parameterizations are possible

    possible alternative parameterizations are

    $x = \cos t^3 \ \ y = \sin t^3$

    $x = \frac{1-t^2}{(1+t^2)^2} \ \ y = \frac{2t}{(1+t^2)^2}$

Specifying Curves

How do we specify a curve? depends on a problem:

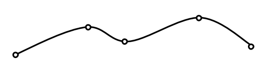
- typically we want a curve to pass through (or at least near) some points; we also want it to be smooth;
- in other situations we may want a very specific curve (e.g. circle for a wheel);
- it can be a solution of an equation, a wavefront …
- we focus on using control points

Before computers

Smooth curves passing through predefined points were used before computers

Draftsmen were using splines (thin flexible strips made out of steel).

When such a strip was fixed at several points, it would acquire a smooth shape,minimizing the bending energy; this shape turns out to be also visually pleasing



Splines

Idea: imitate the real-life splines. It turns out that if we write equations for energy of the plate and minimize the energy, we get piecewise cubic polynomials! (for a particular approximation of the energy).

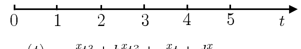
Natural splines:

- piecewise cubic polynomial
- interpolate a set of points
- polynomial segments joined with C2 continuity (first and second derivatives coincide).

Natural splines

Given points $p_i = (x_i, y_i)$, find p.w. cubic polinomial curve interpolating the points, which is differentiable twice

Assume that in parametric domain points



$x_i(t) = a_i^x t^3 + b_i^x t^2 + c_i^x t + d_i^x$

$y_i(t) = a_i^y t^3 + b_i^y t^2 + c_i^y t + d_i^y$

in vector form $p_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i$
segment on $[i, i+1]$

Interpolation condition: $p_i(i+1) = p_{i+1}(i+1) = p_{i+1}$

Continuity of the 1st derivative: $\frac{dp_i}{dt}(i+1) = \frac{dp_{i+1}}{dt}(i+1)$

Continuity of the 2nd derivative: $\frac{dp_i^2}{dt^2}(i+1) = \frac{d^2 p_{i+1}}{dt^2}(i+1)$

Natural splines

Expanding the equations we get

Interpolation condition:

$$a_i(i+1)^3 + b_i(i+1)^2 + c_i(i+1) + d_i = p_{i+1}$$

$$a_{i+1}(i+1)^3 + b_{i+1}(i+1)^2 + c_{i+1}(i+1) + d_{i+1} = p_{i+1}$$

Continuity of the 1st derivative:

$$3(a_i - a_{i+1})(i+1)^2 + 2(b_i - b_{i+1})(i+1) + (c_i - c_{i+1}) = 0$$

Continuity of the 2nd derivative:

$$6(a_i - a_{i+1})(i+1) + 2(b_i - b_{i+1}) = 0$$

For all internal points i we have 4 equations; total of $4(N\text{-}1)$, for N points. We add 2 interpolating conditions on endpoints, and 2 requirements that 2nd derivatives are 0 at endpoints. We get a complete linear system with $4N$ unknowns

Other splines

Local control: **it is convenient if each control point influences only a finite part of the curve; this is not true for natural splines: each control point affects the whole curve.**

Problem**: we want 1st and 2nd derivative continuity, interpolation of control points and local control. We cannot have all with cubic polynomials.**

Why not increase the degree? **High-degree polynomials often are are wiggly. Cubics are the most natural.**

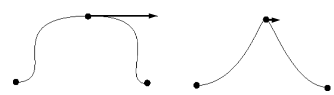
Choices: give up second derivative continuity or give up interpolation.

Interpolating cubic splines

Choice 1: give up 2nd derivative continuity.

- **To define the spline completely, we can specify the 1st derivative at each control point.**
- **The result is ok, but not very good. Also, more difficult to specify:**
- **need to figure out derivative magnitudes, tangent directions**

Interpolating cubic splines



Identical configurations of points with different prescribed tangents for the middle control point; the magnitude is difficult to get right.

This particular type of interpolating C1-continuous cubic splines is called Hermite (there are others, e.g. Catmull-Rom).

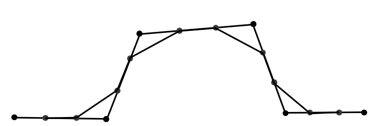
B-splines

Choice 2: give up interpolation of control points.

- **A better choice for interactive design.**
- **The resulting piecewise polynomial curves are *B-spline curves.* B-splines can have any order; most useful are quadratic and cubic.**
- **B-splines can be computed using efficient iterative algorithms (subdivision).**

Subdivision for quadratic splines

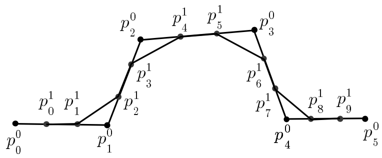
Chaikin's algorithm:



divide each edge into three equal parts;
insert two new (red) points on each edge;
remove old (black) points;
connect new (red) points;
repeat until sufficiently fine subdivision is obtained.

The limit curve is piecewise quadratic!

Subdivision for quadratic splines

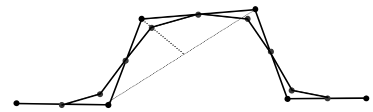


two rules, for odd and even points:

$$p_{2i}^{j+1} = \tfrac{3}{4}p_i^j + \tfrac{1}{4}p_{i+1}^j$$

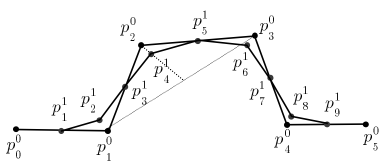
$$p_{2i+1}^{j+1} = \tfrac{1}{4}p_i^j + \tfrac{3}{4}p_{i+1}^j$$

Subdivision for cubic splines



Split each edge in two, insert a new (red) vertex; move old (black) vertices to new positions; connect all vertices.

Subdivision for cubic splines

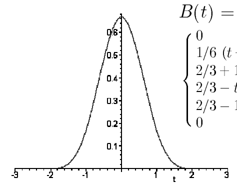


$$p_{2i}^{j+1} = \tfrac{1}{8}p_{i-1}^j + \tfrac{3}{4}p_i^j + \tfrac{1}{8}p_{i+1}^j$$

$$p_{2i+1}^{j+1} = \tfrac{1}{2}p_i^j + \tfrac{1}{2}p_{i+1}^j$$

Basis functions for cubic splines

Uniquely defined by the following conditions:
- **piecewise cubic on integer intervals;**
- **continuous 2nd derivative;**
- **zero outside [-2,2];**
- **integral is 1.**



$$B(t) = \begin{cases} 0 & t < -2 \\ 1/6\ (t+2)^3 & t < -1 \\ 2/3 + 1/2\,t + 1/2\ (t+1)^2 - 1/2\ (t+1)^3 & t < 0 \\ 2/3 - t^2 + 1/2\,t^3 & t < 1 \\ 2/3 - 1/2\,t + 1/2\ (t-1)^2 - 1/6\ (t-1)^3 & t < 2 \\ 0 & 2 \le t \end{cases}$$

Basis functions for cubic splines

Important properties:

Any C²-continuous piecewise cubic on intervals between integers function can be written as

$$p(t) = \sum_{i=-\infty}^{\infty} p_i B(t-i)$$

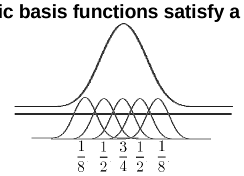
where p_i are the B-spline control points.

Because the basis functions are zero outside a finite interval, each control point influences a finite part of the curve.

Because they are C²-continuous, the curves are C²-continuous

Two-scale relation

Cubic basis functions satisfy a two-scale relation:



Let $B^0(t)$ be the orginal scaling function,
$B^1(t) = B^0(2t)$ **be a similar function, scaled by a factor** $\frac{1}{2}$

Two-scale relation:

$$B^0(t) = \frac{1}{8}B^1(t+1) + \frac{1}{2}B^1(t+\frac{1}{2}) + \frac{3}{4}B^1(t)\frac{1}{2}B^1(t-\frac{1}{2}) + \frac{1}{8}B^1(t-1)$$

Subdivision formulas

Observation: control points connected by line segments approximate the curve. If we can replace control points by a new set denser of control points defining the same curve, we get a better approximation.

To do this, replace $p(t) = \sum_{i=-\infty}^{\infty} p_i^0 B^0(t-i)$ **with**

$$p(t) = \sum_{i=-\infty}^{\infty} p_i^1 B^1(t-i/2)$$

Note that that interval influenced by each new control point p_i^1 **is two times shorter.**

Subdivision formulas

To get expressions for the new control points, substitute the two-scale relation into the formula for the spline and rearrange:

$$p(t) = \sum_i p_i^0 \left(\frac{1}{8} B^1(t-i+1) + \frac{1}{2} B^1(t-i+\frac{1}{2}) + \frac{3}{4} B^1(t-i) \frac{1}{2} B^1(t-i-\frac{1}{2}) + \frac{1}{8} B^1(t-i-1) \right)$$

$$p(t) = \sum_i \underbrace{\left(\frac{1}{8} p_{i-1}^0 \frac{3}{4} p_i^0 + \frac{1}{8} p_{i+1}^0 \right)}_{\text{new even control points}} B(t-i) + \sum_i \underbrace{\left(\frac{1}{2} p_i^0 + \frac{1}{2} p_{i+1}^0 \right)}_{\text{new odd control points}} B(t-i-\frac{1}{2})$$