Mathematical tools 1

Session 2

Franck HÉTROY

M2R IVR, October 12th 2006

First session reminder

- Motivation: interpolate or approximate an ordered list of 2D points P_i
- Definition: spline curve: $C(t) = \sum_{i=0}^{n} F_i(t) P_i$
- Interesting properties:
  - Normality: $\forall t, \sum_{i=0}^{n} F_i(t) = 1$ (affine/barycentric invariance)
  - Positivity: $\forall t, \forall i, F_i(t) \geq 0$ (convex hull)
  - Regularity: $\forall i$, F_i has a single max (oscillation regularization)
  - Locality: $\forall i$, F_i has compact support
  - Parametric/geometric continuity

Interpolation and approximation

1. Splines
   - Basic definitions
   - Splines: a gallery
2. Wavelets and multiresolution

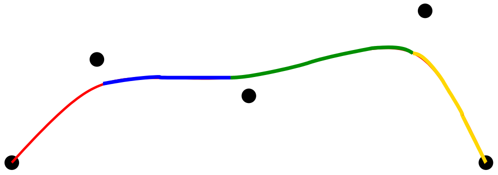
Decomposing a spline

We will look for uniform splines:

$$C(t) = \sum_{i=0}^{n} F_i(t) P_i = \sum_{i=0}^{n-1} C_i(n.t - i)$$

with $C_i : [0, 1] \to \mathbb{R}^3$ polynomial in t

$\Rightarrow$ each C_i corresponds to a curve segment, defined for $t \in [\frac{i}{n}, \frac{i+1}{n}] = [t_i, t_{i+1}]$



Interpolation splines

Which parametric continuity ?

- $C^0 \Rightarrow$ control polygon !
- C^1: shared derivatives D_i

$\Rightarrow$ 4 conditions for each curve segment:

$$\begin{cases} C_i(0) & = & P_i \\ C_i(1) & = & P_{i+1} \\ C_i'(0) & = & D_i \\ C_i'(1) & = & D_{i+1} \end{cases}$$

$\Rightarrow C_i =$ degree 3 polynomial:

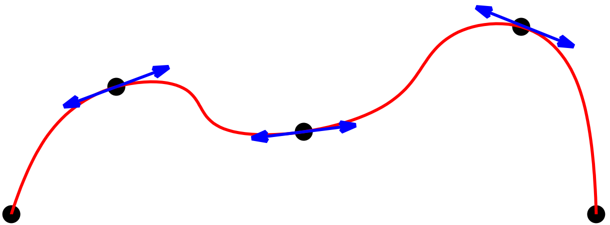
$$C_i(t) = A + Bt + Ct^2 + Dt^3$$

Cubic Hermite splines (first order)

If D_i are given, $C_i(t)$ is uniquely determined:

$$\begin{cases} C_i(t) & = & H_0(t)P_i + H_1(t)P_{i+1} + H_2(t)D_i + H_3(t)D_{i+1} \\ H_0(t) & = & 1 - 3t^2 + 2t^3 \\ H_1(t) & = & 3t^2 - 2t^3 \\ H_2(t) & = & t - 2t^2 + t^3 \\ H_3(t) & = & -t^2 + t^3 \end{cases}$$

Problem: how do we compute D_i values ?



Cubic Hermite splines (second order)

First idea: reinforce continuity $\Rightarrow$ C^2 continuity

$\rightsquigarrow$ conditions:

$$\begin{cases} C_i(0) & = & P_i \\ C_i(1) & = & P_{i+1} \\ C_i'(1) & = & C_{i+1}'(0) \\ C_i''(1) & = & C_{i+1}''(0) \end{cases}$$

$\Rightarrow$ if we have $m+1$ points P_i (i.e. m curve segments), we have $4m$ unknown values (A, B, C, D for each C_i) and $4(m-1)+2$ equations (only 2 for the last segment)
$\Rightarrow$ we need 2 other conditions

Usually, we set

$$\begin{cases} C_0''(0) & = & 0 \\ C_{n-1}''(1) & = & 0 \end{cases}$$

Hermite splines (second order)

Property

For a Hermite spline of order 2, the D_i values are given by:

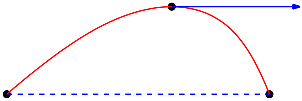
$$\begin{pmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & & \dots & & & \\ & & & \dots & & \\ & & & 1 & 4 & 1 \\ & & & & 2 & 1 \end{pmatrix} \begin{pmatrix} D_0 \\ \dots \\ \dots \\ \dots \\ \dots \\ D_n \end{pmatrix} = \begin{pmatrix} 3(P_1 - P_0) \\ 3(P_2 - P_0) \\ \dots \\ \dots \\ 3(P_n - P_{n-2}) \\ 3(P_n - P_{n-1}) \end{pmatrix}$$

Proof.

4 equations for each $C_i \Rightarrow C_i$ is degree 3. Let $C_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$. Solve the previous equations. □

Cardinal spline

Other idea: we want D_i parallel to $P_{i-1}P_{i+1}$ (no condition on second derivative)



$\rightsquigarrow$ conditions:

$$\left\{\begin{array}{lcl} C_i(0) & = & P_i \\ C_i(1) & = & P_{i+1} \\ C_i'(1) & = & C_{i+1}'(0) \\ C_i'(0) & = & k(P_{i+1}-P_{i-1}) \end{array}\right.$$

(same k for all derivatives)

$\Rightarrow$ each segment curve depends on 4 points:

$$P_{i-1}, P_i, P_{i+1} \text{ and } P_{i+2}$$

Cardinal spline

Property

$$C_i(t) = (t^3\ t^2\ t\ 1)M_{card} \begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix}$$

with

$$M_{card} = \begin{pmatrix} -k & 2-k & -2+k & k \\ 2k & -3+k & 3-2k & -k \\ -k & 0 & k & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Property

A cardinal spline is normal, regular, local of order 4, and C^1-continous.

Approximation splines

Historical background:

- early 1960s: first “numerically-controlled machines” in automotive industry
  $\Rightarrow$ need to design (smooth) curves and surfaces starting from a very few number of 3D points (esp. for coachbuilding)
- Pierre Bézier (Renault, 1960-1963): theoretical study and tests, conception of a whole system (“UNISURF”)
- Paul de Faget de Casteljau (Citroën, 1958-1959): geometric algorithm
- Robin Forrest (Univ. Cambridge, 1972): link between both, popularization

$\rightsquigarrow$ to know more, see Christophe Rabut’s webpage:
http://www-gmm.insa-toulouse.fr/∼rabut/bezier/

Bézier curves

Definition (Bézier curve)

$\forall i, F_i(t) = B_i^n(t) = C_n^i t^i (1-t)^{n-i}$ (Bernstein polynomials).
Remember that $n+1$ is the number of control points.

Property

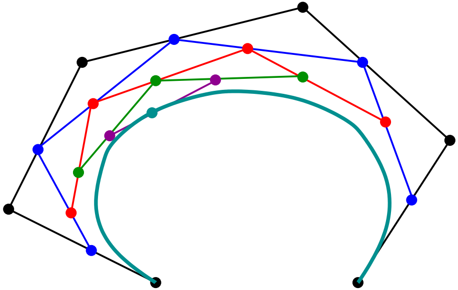
A Bézier curve is normal, positive, regular and C^∞-continuous. Moreover, it goes through first and last points P_0 and P_n.

Problems:

1. a Bézier curve is not local
2. the higher the number of control points, the higher the degree of polynomials
   $\Rightarrow$ computation time and numerical stability problems

de Casteljau's algorithm

- How can we compute points on a Bézier curve without doing all the computations ?
- Idea: divide each segment P_iP_{i+1}, creating a new point $P_i^1 = tP_i + (1-t)P_{i+1}$, then do the same with segments $P_i^1 P_{i+1}^1$, etc.
- Example: $t = 0.4$



Piecewise Bézier curves

Definition (Piecewise Bézier curve)

If $n = 3m + 1$, $C(t) = \sum_{i=0}^{m-1} C_i(m.t - i)$ with
$\forall i \leq m, C_i(t) = B_0^3(t)P_{3i} + B_1^3(t)P_{3i+1} + B_2^3(t)P_{3i+2} + B_3^3(t)P_{3i+3}$.
That is to say,
$C_i(t) = (1-t)^3 P_{3i} + 3(1-t)^2 t P_{3i+1} + 3(1-t)t^2 P_{3i+2} + t^3 P_{3i+3}$.

Property

A piecewise Bézier curve is local of order 2, normal, positive, regular and C^0-continuous. It also goes through points P_{3i}.

Problem: only C^0-continuous

Solution: if $\forall i \leq m, P_{3i+1} = 2P_{3i} - P_{3i-1}$, then C^1-continuous

- That means 1/3 of the points cannot be freely chosen
- If we want C^2-continuity, the curve must be global

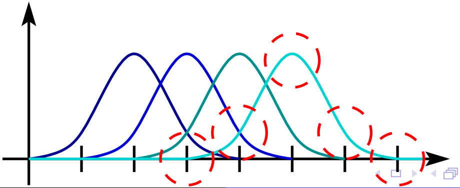
B-splines

Goal: normal, positive, regular, local and C^2-continuous splines

Property

The functions F_i are uniquely determined if we assess:

- *normality*: $\sum_i F_i(t) = 1$
- *locality of order 4*
- *each F_i is made of 4 curve segments, which are cubic polynomials*
- *C^2-continuity between two successive cubic polynomials*



B-splines

Property

$$C_i(t) = (t^3 \; t^2 \; t \; 1) M_{bspline} \begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix}$$

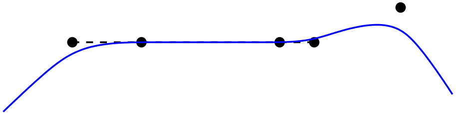
with

$$M_{bspline} = \begin{pmatrix} -1 & 3 & -3 & -1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

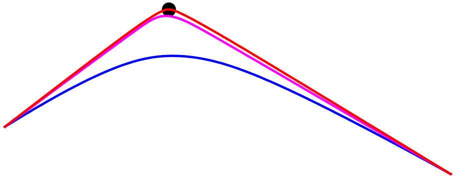
Compare to cardinal splines: pros and cons ?

Particular cases

- If P_{i-1}, P_i, P_{i+1} and P_{i+2} are aligned, then the curve is locally a straight line



- If $P_{i-1} = P_i = P_{i+1}$ (triple point), then this point is interpolated by the curve



- If we want to interpolate first and last points P_0 and P_n, we can add extra points $P_{-1} = 6P_0 - 4P_1 - P_2$ and $P_{n+1} = 6P_n - 4P_{n-1} - P_{n-2}$

B-splines of order k (= degree $k-1$)

Previous were B-splines of order 4/degree 3 (cubic splines)

Definition (B-spline of order k, Cox-de Boor recursion formula)

$F_i = B_{i,k}$, defined by the following recursive formula:

$$B_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{else} \end{cases}$$

$$B_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} B_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(t)$$

with $\forall i = 0 \dots n, t_i \in [0,1]$ and $t_0 \leq t1 \leq \dots \leq t_n$

Property

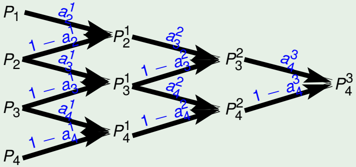
A B-spline of order k is k-local and C^{k-2}-continuous.
Each F_i is made of k curve segments, each of them being a polynomial of degree $k-1$.

de Boor's algorithm

- Generalization of de Casteljau's: construction of points on a B-spline curve
- What's different:
  - weights used to divide each segment $P_i P_{i+1}$ vary;
  - not all control points are involved, only $k + 1$
- Weights are found with a triangular scheme using Cox-de Boor recursion formula

Example

Cubic B-spline, 11 control points ($n = 10$), $\forall i, t_i = i/10$.
$t = 0.45$: we want to compute $C(0.45)$.
Cubic $\Rightarrow$ 4-local $\Rightarrow$ only P_1, P_2, P_3, P_4 involved



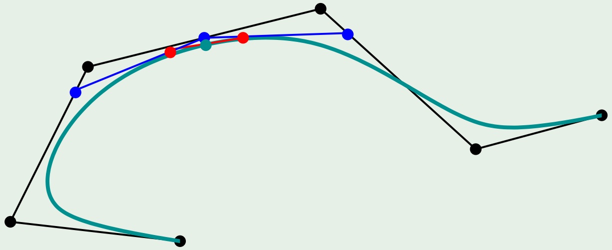
de Boor's algorithm

Property

$$a_i^j = \frac{t_{i+k-j} - t}{t_{i+k-j} - t_i}$$

Example

Here, $a_2^1 = \frac{1}{6}, a_3^1 = \frac{1}{2}, a_4^1 = \frac{5}{6}, a_3^2 = \frac{1}{4}, a_4^2 = \frac{3}{4}, a_4^3 = \frac{1}{2}$.



Beta-splines

Definition (Beta-spline)

$$C_i(t) = (t^3\ t^2\ t\ 1) M_{beta}(\beta_1, \beta_2) \begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix}$$

- Generalization of B-splines
- G^{k-2}-continuous instead of C^{k-2}
- Two parameters β_1 and β_2: bias and tension, to control slope and curvature

N.U.R.B.S.

Problem: with these splines we cannot draw some simple curves (e.g. conics, even a circle !)

$\rightsquigarrow$ Piecewise polynomials as influence functions are not adequate

Example

Circle $C((0,0),1)$: $C(t) = (x(t), y(t))$
$\Rightarrow$ cannot be represented using polynomials, otherwise $x(t)^2 + y(t)^2 = 1$ is a polynomial in t
$\rightsquigarrow C(t) = (\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$

Idea: rather use *rational* polynomials

N.U.R.B.S.

Definition (N.U.R.B.S. of order k)

$$F_i(t) = R_{i,k}(t) = \frac{w_i B_{i,k}(t)}{\sum_{j=1}^{n} w_j B_{j,k}(t)},$$

with w_i real numbers (weights, usually ≥ 0 – what happens if $w_i = 0$?) and $B_{i,k}$ the influence functions of the B-spline of order k having the same control points

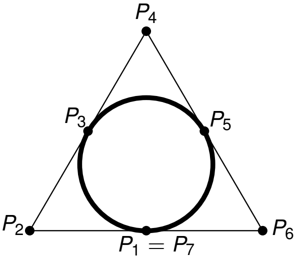
- N.U.R.B.S. stands for *Non Uniform Rational B-Spline*
- B-splines are a special case of N.U.R.B.S. : $\forall i, w_i = 1$
- $R_{i,k}$ are rational polynomials of degree $k - 1$

N.U.R.B.S.

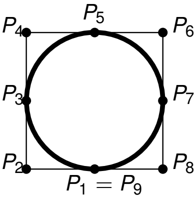
Property

A N.U.R.B.S. curve of order k is normal, positive, regular, k-local and C^{k-2}-continuous.

Circle:



$$w_{2i+1} = 1$$
$$w_{2i} = \frac{1}{2}$$



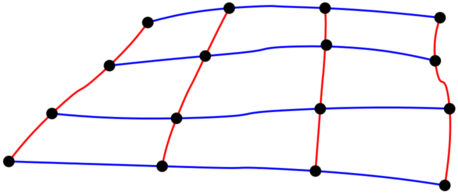
$$w_{2i+1} = 1$$
$$w_{2i} = \frac{\sqrt{2}}{2}$$

Spline surfaces

- Interpolate or approximate a grid of 3D points $P_{i,j}$
- Spline surfaces are constructed by tensor product of spline curves:

$$S(t,t') = \sum_{i=0}^{n} \sum_{j=0}^{m} F_{i,j}(t,t') P_{i,j}$$

- If $F_{i,j}(t,t') = F_i(t)F_j(t')$, then isocurves ($t = cst$ or $t' = cst$) are spline curves
- Same properties as spline curves: normality, regularity, locality, continuity between patches, ...



See you next week

The end !

Interpolation and approximation