Shape Description
using B-Spline Modeling

Irene Gu

November 2000

Object Description

- Boundary or surface-based
  eg: contours, edges, surface
- Region-based
  e.g: color, texture
- Model-based

Boundary-based descriptors

- Fourier descriptor
- B-spline
- Wavelet descriptor
- Polygon
- AR model
- Moment
- Curvature
- Points, lines
- HMM model
- Active contour (snakes)

...

Outlines

•.Categories in object description

• B-spline as an object descriptor

Polynomial spline

B-spline

• Object recognition

• Matlab spline toolbox

• Demons:
Curve / surface modeling and approximation

Local/Global descriptors

Local descriptors:
• change parameters influences local shape
• suitable to occluded and deformed objects
e.g. B-spline, polygon

Global descriptors:
• modify parameters changes global shape
e.g. Fourier descriptor, moments,
AR model

Robustness of descriptors

- affine invariant
  e.g: translation
  rotation
  scaling
  orthographic transformation
  (semi-)perspective projection
- multiscale representation
  e.g: wavelet descriptors
- compression
  e.g: using a few parameters

Polynomial Spline

A spline of order k:

- Piecewise polynomial of degree (k-1);
- Derivatives up to order (k-2)
  at the joints between segments

e.g. Cubic spline: k=4,
polynomial degree =3,
C^2 continuity at the joints (breaks)

Object shape modeling by B-spline

Advantages
- local shape controllability
- continuous curve modeling
- small number of control points
- smoothing curve
- easy to calculate curve derivatives

Disadvantages
- sensitive to initial point selection

Polynomial Spline (con't)

For each segment j
=> continuous cubic spline function:

$$r_j(t) = \sum_{i=1}^{4} B_{i,j} t^{i-1} = B_{1,j} + B_{2,j} t + B_{3,j} t^2 + B_{4,j} t^3$$

$$t_j \le t \le t_{j+1}$$

Given:
- position vector r_j , $1 \le j \le n$
- 1st order derivative at boundary points $r_1^{'}, r_n^{'}$
- Continuity of 2nd derivatives at internal points

$$r_j^{''}(t_{j+1}) = r_{j+1}^{''}(0)$$

Compute polynomial spline

1. Compute segment boundary points

   either by chord length approximation
   or by normalized segments

2. Compute r'_j j=2...n-1

3. Compute $B_{k,j}$ k=1..4, j=1...n

4. Compute (cubic spline)

$$r_j(t) = \sum_{i=1}^{4} B_{i,j} t^{i-1} = B_{1,j} + B_{2,j} t + B_{3,j} t^2 + B_{4,j} t^3$$

j=1,2...n-1

B-spline curve modeling

Given a discrete closed curve

$$r_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad i = 0,1,\ldots n$$

Continuous B-spline curve: (closed curve)

$$r(t) = \sum_{j=0}^{n} r_j(t - t_j) \qquad t \in [t_0, t_n]$$

where $r_j(t)$ is only defined in $t \in [t_j, \ t_{j+1}]$

r(t) consists of (n+1) segments:

knot point: $\hat{r}_j = r_j(t_j)$
connecting point between segments
knot t_j (curve parameter):

Polynomial spline and B-spline

Any polynomial spline ⇔
weighted sum of shifted B-spline basis functions

B-spline:

$$r_j(t) = \sum_{i=0}^{k-1} \tilde{B}_i N_{i,k}(t) \qquad t_j \le t \le t_{j+1}$$

$\tilde{B}_i$: polygon vertics / B-spline coef.

$N_{i,k}$: B-spline basis functions

For one curve segment j

$$r_j(t) = \sum_{i=0}^{k-1} \tilde{B}_i N_{i,k}(t) \qquad t_j \le t \le t_{j+1}$$

$N_{i,k}(t)$: B-spline basis function of order k
(k=4: Cubic B-spline)

Cox-de Boor recursive formulas

$$\begin{cases} N_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{N}_{i,k}(t) = \dfrac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \dfrac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \end{cases}$$

This can be applied to:
open curves with arbitrary spaced knots

For <u>normalized</u> segment $0 \le t < 1$

$$
\begin{cases}
N_{0,4}(t) = \dfrac{-t^3 + 3t^2 - 3t + 1}{6} \\
N_{1,4}(t) = \dfrac{3t^3 - 6t^2 + 4}{6} \\
N_{2,4}(t) = \dfrac{-3t^3 + 3t^2 + 3t + 1}{6} \\
N_{3,4}(t) = \dfrac{t^3}{6}
\end{cases}
$$

$$r_j(t) = N_{0,4}\tilde{B}_j + N_{1,4}\tilde{B}_{j+1} + N_{2,4}\tilde{B}_{j+2} + N_{3,4}\tilde{B}_{j+3}$$

Cubic B-spline curve $r(t) = \sum_{j=0}^{n} r_j(t - t_j)$

$$r(t) = \sum_{j=0}^{n+3} c_{(j+1)\bmod (n+1)} Q_{j,4}(t)$$

$$c_{n+1} = c_0,\ c_{-1} = c_n$$

Bases: $\{\, Q_{j,4}(t) \quad j = 0,1,\ldots.n+3 \}$

=> $Q_{j,4}(t) = Q_{j-1,4}(t-1) = Q_{j+1,4}(t+1)$

set $t' = t - t_j, \quad t \in [\, j-1, j+3]$ =>

$$Q_{j,4}(t) = N_{i,4}(t' - j - 2) + N_{1,4}(t' - j - 1) + N_{2,4}(t' - j) + N_{3,4}(t' - j + 1)$$

control points $\{c_j\}$ ⇔ knot points $\{\hat{r}_j\}$

For normalized cubic B-spline curve:

$$\hat{r}_j = \frac{1}{6} c_{j-1} + \frac{2}{3} c_j + \frac{1}{6} c_{j+1} \qquad j = 0,1,\cdots n$$

Equivalent matrix expression

$$r_j(t) = \frac{1}{6} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 3 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{B}_j \\ \tilde{B}_{j+1} \\ \tilde{B}_{j+2} \\ \tilde{B}_{j+3} \end{bmatrix}$$

$$r_j(t) = T\mathbf{N}\tilde{B}$$

Property: Affine invariance

General affine transform

if $\tilde{r}(t) = \mathbf{A} r(t) + b$

$$\begin{bmatrix} \tilde{x}(t) \\ \tilde{y}(t) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

=> $\tilde{c}_j = \mathbf{A} c_j + b \qquad \mathrm{j} = 0,1,\cdots \mathrm{n}$

special cases:

if $\tilde{r}(t) = \beta\ \mathbf{R}(\theta) r(t) + b$

=> $\tilde{c}_j = \beta \mathbf{R}(\theta) c_j + b \qquad \mathrm{j} = 0,1,\cdots \mathrm{n}$

B-spline curve approximation

Given the number of control points m ($m \ll n$), and the order of B-spline k

Find c_j (or $\hat{r}_j$)

such that: $\min_{t_j} \sum_{j=0}^{n} \left\| r\left(t_j\right) - r_j \right\|^2$

where $r\left(t_j\right)$ is either knot or interpolated point, r_j is the discrete sample

Matlab Spline toolbox

pp form:
polynomial spline
p=ppmak(breaks, coefs)

B-form:
B-spline
f=spmak(knots, coefs)

Tensor product for multivariate:

p(x,y)=f(x)g(y)

sp=spmak({knots(x),knots(y)},coefs)

Object recognition

Matching B-spline curves

Affine similar/ dissimilar:
If $\tilde{r}(t)$ is rotated, scaled and translated from *r(t)*, $e^2(\tilde{r}(t), r(t))$ must be small.

Criterion:

$$e^2(\tilde{r}(t), r(t)) = \min_{\theta,\beta,b} \frac{1}{n} \sum_{j=1}^{n} \left\| \tilde{r}(t_j) - \beta\, \mathbf{R}(\theta)\, r(t_j) - b \right\|^2$$

Algorithm:

- estimate b
- update $\theta_{k+1}, \beta_{k+1}$ using gradient method

$$\theta_{k+1} = \theta_k - \mu_1 \nabla_{\theta_k} \qquad \beta_{k+1} = \beta_k - \mu_2 \nabla_{\beta_k}$$

- update e^2_{k+1} using new $\theta_{k+1}, \beta_{k+1}$ until it converges