1994

Sparse Smooth Connection between Bezier/ Bspline Curves

Chandrajit Bajaj

Guoliang Xu

Report Number:
94-045

https://docs.lib.purdue.edu/cstech/1145

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
Sparse Smooth Connection between Bézier/Bspline Curves

Chandrajit Bajaj and Guoliang Xu
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-94-045
June, 1994
Sparse Smooth Connection between Bézier/Bspline Curves

Chandrajit Bajaj  
Department of Computer Science  
Purdue University  
West Lafayette, IN 47907-1398, USA  
bajaj@cs.purdue.edu

Guoliang Xu  
Department of Computer Science  
Purdue University  
West Lafayette, IN 47907-1398, USA  
xuguo@cs.purdue.edu

June 20, 1994

Introduction

Often in interactive font design, free-form sketching and input path specification for graphics animation, one is faced with the problem of connecting two Bézier or Bspline polynomial curves with a smooth, piecewise transition polynomial curve achieving prescribed continuity at the two end points. Furthermore one desires the transition polynomial curve to have the fewest number of pieces. In this paper we address this issue by solving the following two problems:

Smooth Connection problem: Given two polynomials $P: [a, b] \rightarrow \mathbb{R}$ and $Q: [c, d] \rightarrow \mathbb{R}$ of degree $n$ with $b < c$. Find a piecewise polynomial $R: [b, c] \rightarrow \mathbb{R}$ of degree $n$, such that (1°) $R$ is $C^{\mu}$ continuous in $(b, c)$ for a given integer $\mu$ with $1 \leq \mu \leq n$. (2°) $P$ and $R$ join at $b$ with $C^{\mu_1}$ continuity for a given integer $\mu_1$ with $1 \leq \mu_1 \leq n$. (3°) $R$ and $Q$ joint at $c$ with $C^{\mu_2}$ continuity for a given integer $\mu_2$ with $1 \leq \mu_2 \leq n$.

Sparse Smooth Connection problem: In addition to the above conditions (1°), (2°) and (3°), we require that (4°) $R$ has the fewest number of segments.

As an example, the smooth composite function $(P, R, Q)$ may be a single B-spline. It is obvious that there are possibly infinite ways to join any two polynomials with prescribed continuity. Our goal here is not only to achieve a smooth join but also to make the join as simple as possible. By simple we mean that the polynomial $R$ is to be determined, as far as possible, from $P$ and $Q$.

The solution to both the above problems are derived by the use of blossoming (see [2, 3]). For a given degree $n$ polynomial $F: \mathbb{R} \rightarrow \mathbb{R}$, the blossom of $F$, denoted as $B(F)$, is an $n$-affine symmetric function satisfying $B(t_1, \ldots, t_n)$ is an $n$-affine symmetric function satisfying $B(t_1, \ldots, t_n) = F(t_1)$. A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is called affine if it preserves affine combinations, that is, if $f$ satisfies $f(\sum_i a_i u_i) = \sum_i a_i f(u_i)$ for all real numbers $a_1, \ldots, a_k, u_1, \ldots, u_k \in \mathbb{R}$ with $\sum_i a_i = 1$. A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is called $n$-affine if it is an affine function on each individual argument with the others held fixed. Finally, a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is called symmetric if $f$ keeps its value under any permutation of its arguments.

Solution of the Smooth Connection Problem

Lemma 1. Let $AS_n$ be the set of all $n$-affine symmetric functions, $t_1 \leq \ldots \leq t_n < t_{n+1} \leq \ldots \leq t_{2n}$. Then the map $M: f \in AS_n \rightarrow \{f(t_{i}, t_{i+1}, \ldots, t_{i+n-1})\}_{i=1}^{n+1} \in \mathbb{R}^{n+1}$ is a one to one map between $AS_n$ and $\mathbb{R}^{n+1}$.

Proof: It is obvious that $M$ is a linear map and by $M(f) = (0, \ldots, 0)$ we can prove that $f = 0$. In fact, by the progressive de Casteljau algorithm (see [2]), we have $f(x_1, \ldots, x_n) = 0$ i.e., $f = 0$. Now, the only thing left to be proved is that $M$ is invertible, that is, given $(b_1, b_2, \ldots, b_{n+1}) \in \mathbb{R}^{n+1}$, there exist a $f \in AS_n$ such that...
\[ M(f) = (b_1, \ldots, b_{n+1}). \] This \( f = f^i_i \) can be constructed by the following progressive de Casteljau algorithm:

\[
\begin{align*}
M_G^i &= b_i, \quad i = 1, 2, \ldots, n + 1 \\
M_G^i(x_1, \ldots, x_r) &= \frac{t_1 - x_r}{t_{n+1-r} - x_r} M_G^{i-1}(x_1, \ldots, x_{r-1}) + \frac{x_r - t_{n+1-r}}{t_{n+1-r} - x_r} M_G^{i-1}(x_1, \ldots, x_{r-1}), \quad i = 1, 2, \ldots, n + 1 - r
\end{align*}
\]

for \( r = 1, \ldots, n \) (see Theorem 7.1 of [2]).

**Lemma 2.** Smooth connection problem always has solution.

**Proof:** We prove this lemma via a constructive approach.

(i) If \( n + 1 \leq \mu_1 + \mu_2 \), then the piecewise polynomial \( R \) to be determined degenerates to a single segment, and \( R \) can be determined by using the Hermite interpolation conditions:

\[
\begin{align*}
R^{(i)}(b) &= P^{(i)}(b), \quad i = 0, 1, \ldots, n - \mu_1 \\
R^{(i)}(c) &= Q^{(i)}(c), \quad i = 0, 1, \ldots, n - \mu_2
\end{align*}
\]

If \( n + 1 = \mu_1 + \mu_2 \), the solution is unique. If \( n + 1 < \mu_1 + \mu_2 \), there is no uniqueness. If we let \( R \) to be degree \( 2n - \mu_1 - \mu_2 + 1 \) (\( < n \)). Then we have uniqueness.

(ii) If \( n + 1 > \mu_1 + \mu_2 \), equation (1) has no solution in general. Here we construct a B-spline \( F(x) : [a, d] \rightarrow \mathbb{R} \) such that

\[
F(x)|_{[a, b]} = P(x), \quad F(x)|_{[c, \infty)} = Q(x)
\]

and \( R(x) = F(x)|_{[b, c]} \) satisfies the conditions (1º), (2º) and (3º). Let

\[
T = (t_0 = \ldots = t_n < t_{n+1} = \ldots = t_{n+\mu_1} < t_{n+\mu_2 + 1} \leq \ldots \leq t_{2n-\mu_2 + 1} < t_{2n} < \ldots = t_{2n+2})
\]

where \( t_n = a, \ t_{n+1} = b, \ t_{n+\mu_1 + 1} = c, \ t_{2n} = d \) and \( t_{n+\mu_1 + 1}, \ldots, t_{2n-\mu_2 + 1} \) are chosen such that each of them has multiplicity \( \leq \mu \) in \( T \). Let \( \{N_t^\mu(x)\}_{t=0}^{2n+1} \) be the normalized B-spline bases over \( T \) and let

\[
\begin{align*}
d_t &= f_1(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = 0, 1, \ldots, n \\
d_t &= f_2(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = n + 1, \ldots, 2n + 1
\end{align*}
\]

where \( f_1 = B(P), \ f_2 = B(Q) \) are the blossoms of \( P \) and \( Q \), respectively. Then \( F(x) = \sum_{t=0}^{2n+1} d_t N_t^\mu(x) \) is the required B-spline (see Theorem 3.4 of [3]). In fact, \( F(x) \) is \( C^{n-\mu} \) and \( C^{n-\mu} \) continuous at \( b \) and \( c \) respectively, since \( b \) has multiplicity \( \mu_1 \) and \( c \) has multiplicity \( \mu_2 \). Furthermore, since \( t_{n+\mu_1 + 1}, \ldots, t_{2n-\mu_2 + 1} \) have multiplicity \( \leq \mu \), \( f(x) \) is \( C^{n-\mu} \) continuous on \( (b, c) \). Now we only need to show that condition (2) is satisfied. From Theorem 3.4 of [3], we have

\[
\begin{align*}
d_t &= B(F|_{[a, b]})(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = 0, 1, \ldots, n \\
d_t &= B(F|_{[c, \infty)})(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = n + 1, \ldots, 2n + 1
\end{align*}
\]

Hence

\[
\begin{align*}
f_1(t_{\ell+1}, \ldots, t_{\ell+n}) &= B(F|_{[a, b]})(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = 0, 1, \ldots, n \\
f_2(t_{\ell+1}, \ldots, t_{\ell+n}) &= B(F|_{[c, \infty)})(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = n + 1, \ldots, 2n + 1
\end{align*}
\]

Since \( f_1, f_2, B(F|_{[a, b]}) \) and \( B(F|_{[c, \infty)}) \) are in \( AS_n \), it follows from Lemma 1 that \( f_1 = B(F|_{[a, b]}), \ f_2 = B(F|_{[c, \infty]}), \ Q = F|_{[c, \infty]} \), and then \( P = F|_{[a, b]} \) is unique.

\( \diamond \)

In the above proof we insert \( n + 1 - (\mu_1 + \mu_2) \) knots in \( (b, c) \). Hence \( R \) has at most \( n + 2 - (\mu_1 + \mu_2) \) pieces. Since the \( t_{n+\mu_1 + 1}, \ldots, t_{2n-\mu_2 + 1} \) knots can be arbitrarily chosen under the required conditions, \( R \) is not unique. We therefore have the following corollary

**Corollary 3.** The Sparse Smooth Connection problem always has a solution.
The Computation of the Sparse Smooth Connection Polynomial

The proof of Lemma 2 has already provided a way to compute the smooth transition polynomial \( R \). Furthermore this uses only the information that comes from \( P \) and \( Q \) and some inserted knots. However the number of pieces of \( R \) may not be minimal. In order to get a sparse connection polynomial we intend to insert the least number of knots. As in the discussion above, there are two possible cases.

(i) If \( n + 1 \leq \mu_1 + \mu_2 \), the problem is reduced to Hermite interpolation problem as before. One segment is enough to connect the two given polynomials. Then the number of segments is minimum. Now we give a B-spline representation of the composite function. Let

\[
T = (t_0 = \ldots = t_n < t_{n+1} = \ldots = t_{n+\mu_1} < t_{n+\mu_1+1} = \ldots = t_{2n+\mu_1+\mu_2+1})
\]

and \( \{N^n_l(x)\}_{l=0}^{n+\mu_1+\mu_2} \) be the normalized B-spline bases over \( T \). Then \( F(x) = \sum_{l=0}^{n+\mu_1+\mu_2} d_l N^n_l(x) \) is the required function, where

\[
d_l = f_1(t_{l+1}, \ldots, t_{l+n}), \quad \ell = 0, 1, \ldots, n
\]

\[
d_l \text{ are free,} \quad \ell = n + 1, \ldots, \mu_1 + \mu_2 - 1
\]

\[
d_l = f_2(t_{l+1}, \ldots, t_{l+n}), \quad \ell = \mu_1 + \mu_2, \ldots, n + \mu_1 + \mu_2
\]

(ii) If \( n + 1 > \mu_1 + \mu_2 \), then the computation of the inserted knots proceeds as following, with \( i \) increasing from 0 to \( n + 1 - (\mu_1 + \mu_2) \)

(a) Let

\[
T_i = (t_0 = \ldots = t_n < t_{n+1} = \ldots = t_{n+i} < x_1 \leq \ldots \leq x_i < t_{n+i+1} = \ldots = t_{2n+i+1})
\]

where \( t_n = a, \ t_{n+1} = b, \ t_{n+i+1} = c, \ t_{n+i+1} = d \) and \( x_1, \ldots, x_i \) are the knots to be determined and satisfying the following conditions:

\[
b < x_j < c
\]

\[
x_j \text{ has multiplicity} \leq \mu \text{ in } T_i
\]

(b) For \( \ell = i + \mu_1 + \mu_2, \ldots, n \), the de Boor points (see [3]) \( d_\ell \) are determined satisfying conditions from both \( P \) and \( Q \). These double conditions leads to the following equations for unknowns \( x_1, \ldots, x_i \)

\[
B(P)(t_{l+1}, \ldots, t_{n+i}, x_1, \ldots, x_i, t_{n+i+1}, \ldots, t_{l+n}) = B(Q)(t_{l+1}, \ldots, t_{n+i}, x_1, \ldots, x_i, t_{n+i+1}, \ldots, t_{l+n})
\]

for \( \ell = i + \mu_1 + \mu_2, \ldots, n \), or

\[
ge_l(x_1, \ldots, x_i) := B(P - Q)(t_{l+1}, \ldots, t_{n+i}, x_1, \ldots, x_i, t_{n+i+1}, \ldots, t_{l+n}) = 0
\]

for \( \ell = i + \mu_1 + \mu_2, \ldots, n \). There are \( n + 1 - (i + \mu_1 + \mu_2) \) equations and \( i \) unknowns. The ideal cases (a unique solution is expected) are \( i = n + 1 - (i + \mu_1 + \mu_2) \) or \( i = \frac{n+1-(\mu_1+\mu_2)}{2} \). Comparing with the proof of Lemma 2, in which we insert \( n + 1 - (\mu_1 + \mu_2) \) knots, this ideal case will reduce the number of the inserted knots to half. For example, if \( n = 3 \) (cubic), \( \mu_1 = \mu_2 = 1(C^2 \text{ continuity}) \), then \( i = 1 \). If \( \mu_1 = \mu_2 = 2(C^1 \text{ continuity}) \), then \( i = 0 \). If \( n = 5, \mu_1 = \mu_2 = 2(C^3 \text{ continuity}) \), \( i = 1 \). If \( \mu_1 = \mu_2 = 3(C^4 \text{ continuity}) \), \( i = 2 \).

Let \( P(x) = \sum_{j=0}^{n} a_j x^j, \ Q(x) = \sum_{j=0}^{n} b_j x^j \). Then \( B(P - Q)(u_1, \ldots, u_n) = \sum_{j=0}^{n} (a_j - b_j) f\left( \frac{n}{j} \right) \sigma_{jn}(u_1, \ldots, u) \)

where \( \sigma_{jn}(u_1, \ldots, u_n) \) is the j-th n-variable elementary symmetric function[1]. Therefore \( g_l(x_1, \ldots, x_i) \)
can be written as \( g_t = \sum_{j=0}^{i} a_j^{(t)} \sigma_j(x_1, \ldots, x_i) \). Let \( \sigma_j = \sigma_j(x_1, \ldots, x_i) \) be the unknowns, \( j = 1, 2, \ldots, i \), \( \sigma_0 = 1 \). We thus have the following system of linear equations

\[
\begin{bmatrix}
  a_1^{(t+\mu_1+1)} & a_2^{(t+\mu_1+1)} & \cdots & a_{i+1}^{(t+\mu_1+1)} \\
  \vdots & \vdots & & \vdots \\
  a_1^{(n)} & a_2^{(n)} & \cdots & a_{i}^{(n)} \\
\end{bmatrix}
\begin{bmatrix}
  \sigma_1 \\
  \sigma_2 \\
  \vdots \\
  \sigma_i \\
\end{bmatrix}
= -
\begin{bmatrix}
  a_0^{(t+\mu_1+1)} \\
  \vdots \\
  a_{i}^{(n)} \\
\end{bmatrix}
\]  

(7)

(c) If Equation (7) has no solution, increase \( i \) by 1, until it has a solution (may have many solutions). Let \([\sigma_1, \ldots, \sigma_i]^T\) be a solution of (7). Form a polynomial equation

\[
h(x) := \sum_{k=0}^{i} (-x)^{i-k} \sigma_k = 0
\]  

(8)

If all the roots \( x_j \) of \( h(x) \) are real, and they satisfy (6), then we get the required knots \( x_j \). Otherwise, we increase \( i \) until the required knots are obtained. If (7) has many solutions, a closed form of the solution of (8) is helpful to get the required solution. If \( i < 5 \), the closed form of the root \( x_j \) are available.

The case \( i = 0 \) needs separate consideration, since the equation (7) and (8) are degenerate. In this case \( g_t \) are constants. If they are all zero, then we do not need to insert knots in \((b, c)\) and the de Boor points are computed by (4), but no degree of freedom is left. If not all \( g_t \) are zero, we need to consider the next \( i \).

Since we wish to find the solution \( x_j \)'s that satisfy condition (6), we solve Equation (8) for \( \sigma_k \) that satisfies the following necessary condition

\[
\begin{pmatrix}
  i \\
  k 
\end{pmatrix} b^k < \sigma_k < \begin{pmatrix}
  i \\
  k 
\end{pmatrix} c^k, \quad k = 1, 2, \ldots, i
\]  

(9)

(d) Let \( t_{n+\mu_1+j} = x_j \) for \( j = 1, \ldots, i \). Let

\[
d_{\ell} = f_1(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = 0, 1, \ldots, n
\]

\[
d_{\ell} = f_2(t_{\ell+1}, \ldots, t_{\ell+n}), \quad \ell = n+1, \ldots, n+\mu_1+\mu_2+i
\]  

(10)

Then similar to the proof of Lemma 2, we know that the B spline function \( F(x) = \sum_{\ell=0}^{n+\mu_1+\mu_2+i} d_{\ell} N_\ell^n(x) \) is what we require, where \( \{N_\ell^n(x)\}_{\ell=0}^{n+\mu_1+\mu_2+i} \) is the normalized B-spline bases over \( T_i \).

**Psuedocode of the Algorithm**

We present psuedocode for the above algorithm of computing Sparse Smooth Connection polynomials. Here we assume we have (by now standard) library procedures for solving a linear equation and for finding the real roots of a polynomial.

**Sparse Smooth Connection Algorithm**

\( P \) is the input coefficients array of the polynomial \( P \) in power bases

\( Q \) is the input coefficients array of the polynomial \( Q \) in power bases

\( A, B \) are the input end points of interval \([a, b]\)

\( C, D \) are the input end points of interval \([c, d] \)

\( N \) is the degree of the given polynomials

\( MU1 \) is the input continuity at \( b \)

\( MU2 \) is the input continuity at \( c \)
MU is the input continuity in \((b, c)\)

\(D\) is the output coefficients array of the de Boor points \(d_i\)

Knots is the output inserted knots in \((b, c)\)
1 is the output number of inserted knots

\[ I = 0 \]

compute the blossoming of \( P, Q \) and \( P - Q \)

for \( j = 0 \) to \( N \) step 1

\[ P(j) = P(j) / \binom{N}{j} \quad Q(j) = Q(j) / \binom{N}{j} \quad C(j) = P(j) - Q(j) \]

next \( j \)

form knots \( T \), see (3)

for \( j = 0 \) to \( 2N + MU1 + MU2 + 1 \) step 1

\[ \text{if } j \leq N \text{ then } T(j) = A \]
\[ \text{else if } j \leq N + MU1 \text{ then } T(j) = B \]
\[ \text{else if } j \leq N + MU1 + MU2 \text{ then } T(j) = C \]
\[ \text{else } T(j) = D \]

end if

next \( j \)

if \( N + 1 \leq MU1 + MU2 \) then

compute \( d_i \), by formulas (4)

for \( I = 0 \) to \( N + MU1 + MU2 \) step 1

for \( j = 1 \) to \( N \) step 1

Point\((j) = T(I+j) \)

next \( j \)

\[ \text{if } I \leq N \text{ then } \]
\[ \text{call EVALUATE}(P, N, \text{Point}, N, \text{Coeffout}) \]
\[ D(I) = \text{Coeffout}(0) \]
\[ \text{else if } I \geq MU1 + MU2 \text{ then } \]
\[ \text{call EVALUATE}(Q, N, \text{Point}, N, \text{Coeffout}) \]
\[ D(I) = \text{Coeffout}(0) \]
\[ \text{else } D(I) \text{ are free, set to zero} \]

end if

next \( I \)

else

for \( i = 1 \) to \( N + 1 - (MU1 + MU2) \) step 1

for \( I = i + MU1 + MU2 \) to \( N \) step 1

Point\((j) = T(I+j) \)

next \( j \)

\[ \text{call EVALUATE}(C, N, \text{Point}, N-i, \text{Coeffout}) \]
\[ \text{for } k = 1 \text{ to } i \text{ step 1 } \]
\[ \text{Matrix}(I - i - MU1 - MU2, k - 1) = \text{Coeffout}(k) \]
\[ \text{next } k \]
\[ \text{LeftHand}(I - i - MU1 - MU2) = - \text{Coeffout}(0) \]
\[ \text{next } I \]

\[ \text{call LINEARSOLVER}(	ext{Matrix, LeftHand, Solution}) \]
\[ \text{call POLYSOLVER}(	ext{Solution, i, Knots}) \]

if all Knots satisfy the condition (6) then goto L

next \( i \)

form knots \( T_i \), see (6)

L: \( I = 1 \)

for \( j = 0 \) to \( 2N + MU1 + MU2 + i + 1 \) step 1

\[ \text{if } j \leq N \text{ then } T(j) = A \]
\[ \text{else if } j \leq N + MU1 \text{ then } T(j) = B \]
\[ \text{else if } j \leq N + MU1 + i \text{ then } T(j) = \text{Knots}(j - N - MU1) \]
\[ \text{else if } j \leq N + MU1 + MU2 + i \text{ then } 6 T(j) = C \]
\[ \text{else } T(j) = D \]

end if

next \( j \)

compute the de Boor point \( d_i \) by (10)

for \( I = 0 \) to \( N \) step 1
Procedure to evaluate an n-affine symmetric function

procedure EVALUATE(Coeffin, N, Point, M, Coeffout)

Coeffin is the input coefficients array
N - 1 is the number of coefficients
Point is the input evaluating points array
M is the number of evaluating points
Coeffout is the output coefficients array

for j = 0 to N step 1
    Coeffout(j) = Coeffin(j)
next j

for k = 0 to M - 1 step 1
    for j = 1 to N - k step 1
        Coeffout(j-1) = Coeffout(j-1) + Point(k) * Coeffout(j)
    next j
next k

return

References

