The Visual Computer

Surface approximation to scanned data

Les A. Piegl[1],
Wayne Tiller[2]

[1] Department of Computer Science & Engineering, University of South Florida, 4202 Fowler Avenue, ENG 118, Tampa, FL 33620, USA
[2] GeomWare, Inc., 3035 Great Oak Circle, Tyler, TX 75703, USA
e-mail: piegl@grad.csee.usf.edu, geomware@gower.net

A method to approximate scanned data points with a B-spline surface is presented. The data are assumed to be organized in the form of $Q_{i,j}, i = 0, \ldots, n; j = 0, \ldots, m_i$, i.e., in a row-wise fashion. The method produces a $C^{(p-1,q-1)}$ continuous surface (p and q are the required degrees) that does not deviate from the data by more than a user-specified tolerance. The parametrization of the surface is not affected negatively by the distribution of the points in each row, and it can be influenced by a user-supplied knot vector.

Key words: Data approximation – Reverse engineering – B-splines – Curves and surfaces – Algorithms

Correspondence to: L.A. Piegl

1 Introduction

Surface approximation to rectangularly arranged $n \times m$ points has been an integral part of surface design and modeling in CAD/CAM. Recent advances in reverse engineering (Putambekar et al. 1995; Várady et al. 1997) call for different approximation methods as data obtained from acquisition devices cannot be guaranteed to follow the traditional "point carpet" topology. Surface construction to point-cloud data has been investigated both for visualization (Amenta et al. 1998–Hoppe et al. 1994; Schmitt et al. 1986) as well as for reverse engineering applications (Ma and Kruth 1994–Milroy et al. 1995; Sarkar and Menq 1991; Sapidis and Besl 1995; Tuohy et al. 1997). While it is important to be able to construct surfaces to arbitrary points, object scanning, especially for reverse engineering, is not done in just any random manner. Scanners most frequently scan in an organized fashion, producing rows of data points [a typical example is shown in Várady et al. (1997).] The rows may or may not contain the same number of points, and the distribution of points in each row may vary quite a bit.

In this paper, we investigate the following problem. Given a data set in the form:

$$Q_{i,j} \quad i = 0, \ldots, n \quad j = 0, \ldots, m_i,$$

a degree (p, q) B-spline surface is sought that approximates the data up to a user specified tolerance ϵ. Neither the point set is assumed to have a rectangular topology, nor is it assumed that the points are evenly distributed along each row (Fig. 1a, b shows typical data sets). The method is outlined as follows.

1. Divide the input tolerance into three components; that is, $\epsilon = \epsilon_u + \epsilon_v + \epsilon_k$.
2. Fit curves to the rows of data points so that each curve does not deviate from the points more than ϵ_v.
3. Fit a surface to the rows of curves so that it does not deviate from any curve more than ϵ_u.
4. Remove all removable knots from the surface in step 3 using the tolerance ϵ_k.

There are two key issues in this process: (1) the fitting of curves to each row of data points independently of one another, and (2) avoiding data explosion by proper control of the knot vectors.

The organization of the paper is as follows. In Sect. 2, B-spline notations are presented, and in Sect. 3, the general curve approximation problem is outlined. Section 4 gives details on how to approximate curves to a given tolerance with knot-vector

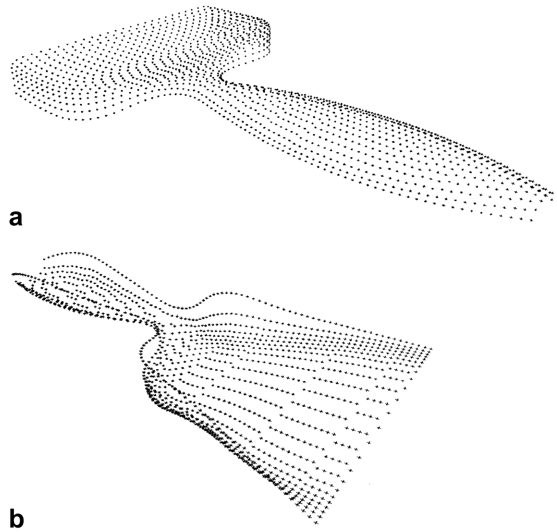


Fig. 1a, b. Data sampled from: **a** a brush handle (1454 points); **b** a dust pan (1229 points)

control. In Sect. 5, the surface approximation method is discussed, and Sect. 6 concludes the paper.

2 B-spline notations

For notational convenience, we introduce curve and surface definitions first. A B-spline curve of degree p is defined as

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i,$$

where P_i are the control points, and $N_{i,p}(u)$ are the normalized B-splines defined over the knot vector

$$U = \{\underbrace{u_0 = \cdots = u_p}_{p+1}, u_{p+1}, \ldots, u_n, \underbrace{u_{m-p} = \cdots = u_m}_{p+1}\}.$$

A B-spline surface of degree (p, q) has the form

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j},$$

where the points $P_{i,j}$ form a control net and the B-splines are defined over

$$U = \{\underbrace{u_0 = \cdots = u_p}_{p+1}, u_{p+1}, \ldots, u_n, \underbrace{u_{r-p} = \cdots = u_r}_{p+1}\}$$

$$V = \{\underbrace{v_0 = \cdots = v_q}_{q+1}, v_{q+1}, \ldots, v_m, \underbrace{v_{s-q} = \cdots = v_s}_{q+1}\}.$$

We assume that the knot vectors U and V are always clamped, i.e., they are in the forms just given, and that the curves and surfaces are nonrational. For more details on B-spline curves and surfaces, the reader is referred to the authors' text (Piegl and Tiller 1997).

3 Curve approximation

In order for the curve approximation to be useful for surface construction, the following issues must be dealt with:

- General curve formulation
- Selection of parameters and knots
- Knot-vector control.

In the subsections that follow, each problem is investigated in detail.

3.1 Least-squares curve approximation

The approximation problem is stated as follows. Given a set of points

$Q_i \quad i = 0, \ldots, k,$

a set of precomputed parameters

$t_i \quad i = 0, \ldots, k,$

and a knot vector U, a degree p B-spline curve with $n+1$ control points is sought that interpolates the end points; i.e.,

$C(t_0) = P_0 = Q_0 \quad C(t_k) = P_n = Q_k,$

and approximates the remaining points in the least-squares sense, that is,

$$\sum_{i=1}^{k-1} |Q_i - C(t_i)|^2$$

is a minimum with respect to the independent variables $P_1, \ldots, P_{n-1}$. The standard least-squares method yields the following system of equations (Piegl and Tiller 1997).

$$\left(N^T N\right) P = R,$$

where $\boldsymbol{N}$ is a $(k-1) \times (n-1)$ matrix

$$\begin{bmatrix} N_{1,p}(t_1) & \cdots & N_{n-1,p}(t_1) \\ \vdots & \ddots & \vdots \\ N_{1,p}(t_{k-1}) & \cdots & N_{n-1,p}(t_{k-1}) \end{bmatrix},$$

$\boldsymbol{R}$ is a vector of $(n-1)$ elements

$$\begin{bmatrix} N_{1,p}(t_1)R_1 + \cdots + N_{1,p}(t_{k-1})R_{k-1} \\ \vdots \\ N_{n-1,p}(t_1)R_1 + \cdots + N_{n-1,p}(t_{k-1})R_{k-1} \end{bmatrix}$$

with

$$R_i = Q_i - N_{0,p}(t_i)Q_0 - N_{n,p}(t_i)Q_k \quad i = 1, \ldots, k-1,$$

and $\boldsymbol{P}$ is the vector or unknown control points

$$\begin{bmatrix} P_1 \\ \vdots \\ P_{n-1} \end{bmatrix}.$$

To solve for the unknown control points, proper parameters and a knot vector are needed.

3.2 Parameters and knot vector for approximation

The parameters should reflect the distribution of the points. Practical experience shows that chordal length parametrization is quite satisfactory in most applications. Let L be the total chordal length

$$L = \sum_{i=1}^{k} |Q_i - Q_{i-1}| .$$

Then

$$t_0 = 0$$

$$t_i = t_{i-1} + \frac{|Q_i - Q_{i-1}|}{L}, \quad i = 1, \ldots, k-1$$

$$t_k = 1.$$

The knot vector is a delicate matter. There are two issues here: (1) the numerical stability of the system of equations, and (2) the quality of the approximating curve. There are many ways to choose a knot vector that gives stable equations. However, in our experience, not all stable equations yield acceptable curves. The method we have chosen is explained as follows.

INPUT:
$t_i, i = 0, \ldots, k$, parameters
p, degree
n, highest index of control points
OUTPUT:
$U = \{u_0, \ldots, u_{n+p+1}\}$, knot vector
ALGORITHM:

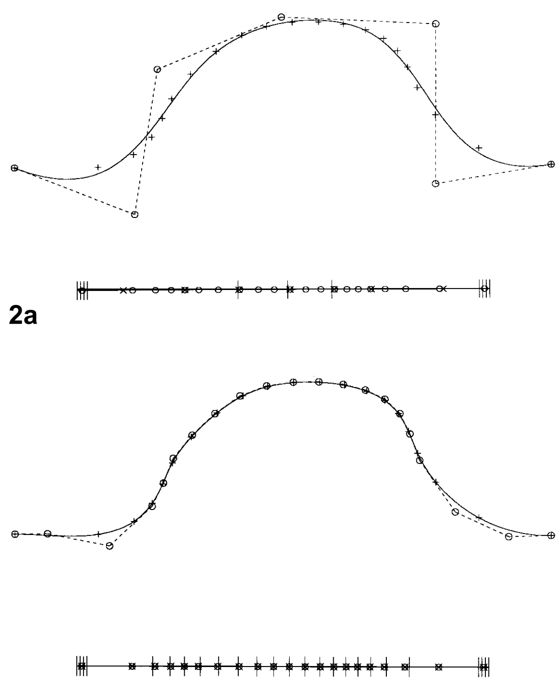
```
for i = 0 to p do   { u_i = t_0; u_{n+i+1} = t_k; }
inc = (k+1)/(n+1); low = high = 0; d = -1;
for i = 0 to n do
    d = d + inc;
    high = ⌊d + 0.5⌋;
    sum = 0.0;
    for j = low to high do   sum = sum + t_j;
    w_i = sum/(high - low + 1);
    low = high + 1;
end

for i = 1 to n - p do
    sum = 0.0;
    for j = i to i + p - 1 do   sum = sum + w_j;
    u_{i+p} = sum/p;
end
```

This algorithm first groups $(k+1)/(n+1)$ consecutive parameters and averages them to have $n+1$ representative values w_i, $i = 0, \ldots, n$. Then p consecutive values are averaged to yield the knots. This choice of the knot vector has two important properties.

1. It gives a stable system of equations with a diagonally dominant coefficient matrix that has semibandwidth less than $p+1$ (Cox 1971; Schoenberg and Whitney 1953).
2. If $n = k$, the approximation problem becomes an interpolation problem, and the resultant knot vector coincides with the one for interpolation (Piegl an Tiller 1997).

Figure 2a, b shows approximation examples. In Fig. 2a, $k = 20$, $n = 6$, and $p = 3$. The circles mark the parameter values, the $\times$'s represent the w_i averages, whereas the bars are the knots. Figure 2b illustrates the interpolation case. Note that because $w_i = t_i$, the interpolation and the approximation knot vectors are the same.

The choice of the knot vector becomes more crucial in approximation up to a given tolerance. Tight tolerances may require that $n \approx k$, and this in turn requires that the knot vector should be close to the


2a
2b

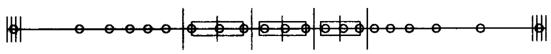

3

Fig. 2a, b. Data approximation: **a** $k = 20$, $n = 6$, $p = 3$; **b** $k = 20$, $n = 20$, $p = 3$

Fig. 3. Knots and their flexibility intervals

one used for interpolation. Otherwise, the approximating curve can be fairly wiggly, especially towards the end points.

3.3 Knot-vector control

Although the choice of the knot vector is crucial in approximation, it turns out that each knot has some flexibility locally. That is, the positions of the knots can be perturbed without causing either numerical problems or creating curves of unacceptable shape. Our idea is to define an interval in which each internal knot can freely be moved. Denoting the knot vector for degree p approximation by U^p, these intervals are defined as

$$u_{i-1}^{p-1} < u_i^p < u_i^{p-1}.$$

That is, the knot for a degree p approximation is bracketed into an interval defined by knots used for a degree $p-1$ approximation. Introducing a percentage parameter per, one can control the width of each interval. In other words, $per = 0\%$ gives no flexibility, whereas for $per = 100\%$, any value within $(u_{i-1}^{p-1}, u_i^{p-1})$ may be chosen. Figure 3 shows an example. The large bars represent knots for the $p = 2$ approximation, the small bars are the knots for the $p = 3$ approximation, and the small rectangles are the flexibility intervals for $per = 75\%$.

The approximation algorithm that uses knot vector control can then be explained as

INPUT:
 $Q_i, i = 0, \ldots, k$, data points
 $\hat{U} = \{\hat{u}_0, \ldots, \hat{u}_{\hat{m}}\}$, candidate knots
 n, p, highest index and required degree
 per, percentage of interval use
OUTPUT:
 $C(u)$, curve approximant
 $\hat{U}$, updated knot vector
ALGORITHM:
 $T = \{t_0, \ldots, t_k\} \leftarrow$ get parametrization;
 $U^p = \{u_0^p, \ldots, u_{n+p+1}^p\} \leftarrow$ knots for degree p;
 $U^{p-1} = \{u_0^{p-1}, \ldots, u_{n+p}^{p-1}\} \leftarrow$ knots for degree $p-1$;
 $S = \{s_0, \ldots, s_{n+p+1}\} \leftarrow$ memory for knots for approximation;
 $s_i \leftarrow t_0, i = 0, \ldots, p$;
 $s_i \leftarrow t_k, i = n+1, \ldots, n+p+1$;
 for $i = p+1$ **to** n **do**
  $a = (1 - per) * u_i^p + per * u_{i-1}^{p-1}$;
  $b = (1 - per) * u_i^p + per * u_i^{p-1}$;
  $\hat{u}_\ell \leftarrow$ input knot closest to u_i^p;
  if($\hat{u}_\ell \in (a, b)$) $s_i = \hat{u}_\ell$ **else** $s_i = u_i^p$;
 end
 $C(u) \leftarrow$ approximate $Q_i, i = 0, \ldots, k$, with T and S;
 $\hat{U} \leftarrow$ **Merge**($\hat{U}, S$);

The algorithm accepts the data points and a knot vector as input. It then sees if the given knots can be used to approximate the data. To do that, it sets up an "ideal" knot vector (the one outlined in Sect. 3.2) for degree p and another one for degree $p-1$. If there are input knots in a flexibility interval, the algorithm selects the one closest to the ideal knot. If no input knot is present in an interval, the ideal knot is used. After curve approximation is achieved, the input knot vector is updated by adding the ideal knots whose flexibility intervals did not contain input knots. Figure 4 shows an example. The small bars are the ideal knots corresponding to the solid curve. The large bars are the ones selected from an input knot vector. They all fall within the flexibility intervals obtained

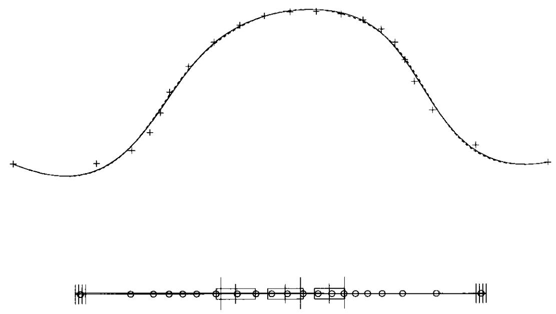


Fig. 4. Approximations with different knots

by *per* $= 75\%$. The corresponding curve is dashed. What is truly interesting to note is that the curves do not deviate much, even though the knots are quite far apart. Our experience shows that *per* $= 100\%$ can safely be applied for almost all practical applications, especially if the data set is dense, which is necessary to justify approximation as opposed to interpolation anyway.

The idea of passing a knot vector into the curve approximator makes sense only if there are more than one approximation, precisely the case used in surface approximation. Each row of data points is independently approximated with a given knot vector passed in for each row and subsequently updated. It is anticipated that after several approximations to given rows, the updated knot vector has enough knots so that no or very few knots must be added. This in turn may allow a tremendous data reduction in the output surface knots, as the independently fitted curves have many knots in common. When they are made compatible, only a few knots must be inserted into each curve.

4 Approximation to a given tolerance

The approximation method discussed in the previous section requires the number of control points as input. In practical applications, what is needed is a curve that approximates the data up to a given tolerance ϵ. Therefore, a fitting routine must determine n so that the perpendicular errors are smaller than the tolerance, that is,

$$\left|Q_i - Q_i^{\pi}\right| < \epsilon \quad i = 1, \ldots , k-1,$$

where Q_i^{π} is the projection of Q_i onto the curve. The following iterative procedure worked well in practice.

1. The input to the routine are the data points Q_i, $i = 0, \ldots , k$, the tolerance ϵ, the required degree p, and a knot vector $\hat{U}$ (Fig. 5a).
2. The output is a curve $C(u)$ and the updated knot vector $\hat{U}$.
3. Compute parameters $T = \{t_0, \ldots , t_k\}$ and initialize an error vector $E = \{e_0, \ldots , e_k\}$ to the zero vector.
4. Interpolate $Q_i, i = 0, \ldots , k$, with $\hat{U}$ passed in (Fig. 5b).
5. Set up knot priorities. Knots passed in receive priority 2, and added knots are assigned priority 1.
6. Remove all removable knots, observing their priorities (Fig. 5c). In other words, try to remove all removable knots with priority 1 first, then the removable ones with priority 2. Details of knot removal are found in Piegl and Tiller (1997). This priority-based knot removal ensures that the algorithm preserves as many input knots as possible.
7. Using the number of control points obtained in the previous step and the input knot vector, approximate the data, applying the algorithm discussed in Sect. 3.3. Project each data point onto the curve and update the error vector E; that is, $e_i = \left|Q_i - Q_i^{\pi}\right|$, where Q_i^{π} is the projection of Q_i onto the curve.
8. If for some index ℓ, $e_\ell > \epsilon$, increase the number of control points and reapproximate. Continue this until $e_i < \epsilon, i = 0, \ldots , k$ (Fig. 5d).
9. Set up knot priorities as already described, pass E to the knot removal routine, and remove knots with priorities until $e_i < \epsilon, i = 0, \ldots , k$, still holds (Fig. 5e).
10. Update the output knot vector by adding the new knots needed by the approximation routine.

A number of comments are in order.

- It is almost impossible to guess the number of control points necessary to approximate the data so that the curve does not deviate from the points more than the tolerance. An initial interpolation followed by knot removal gives a fairly close estimate.
- Starting the iterative process with an approximation as opposed to an interpolation almost always gives better results, hence the use of an approximative curve in step 7. Compare the curves in

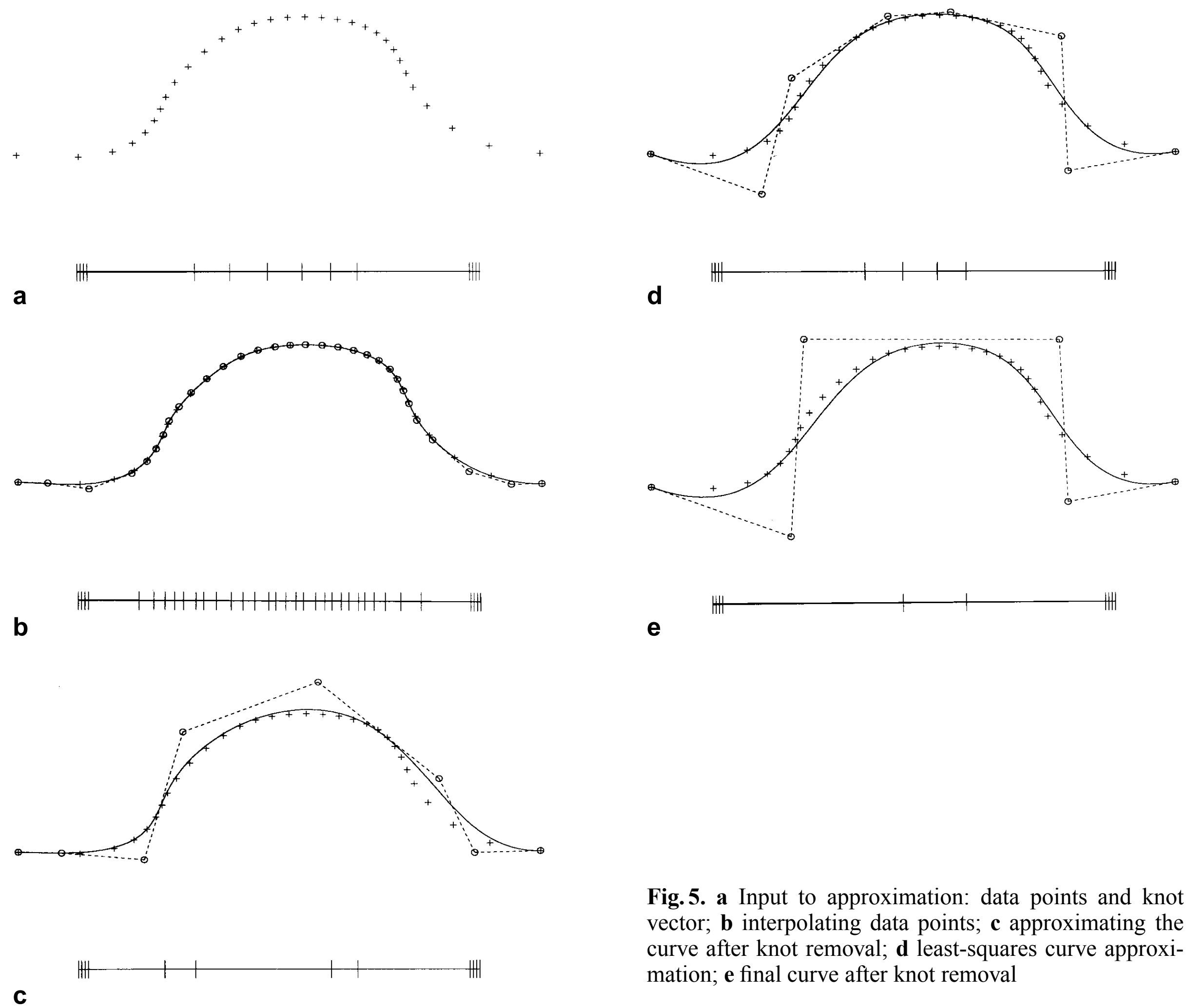


Fig. 5. **a** Input to approximation: data points and knot vector; **b** interpolating data points; **c** approximating the curve after knot removal; **d** least-squares curve approximation; **e** final curve after knot removal

Fig. 5c and d. The error in Fig. 5d is much more evenly distributed than the one in Fig. 5c.

- If the estimate for the number of control points was not adequate, it is increased in the following way.
  - Get the increment $\Delta n = \frac{n}{2}$.
  - Update the index $n = \min\left(n + \Delta n, \frac{n+k+1}{2}\right)$.

  If $n \ll k$, then n is incremented by Δn. However, if $n \approx k$, it is incremented in a binary search fashion.
- The final knot-removal step does not remove many knots in general. However, it cleans up the knot vector by removing knots that happen to lie close to one another.
- Based on our experience, we set $per = 100\%$ and did not use it as a user-adjustable input. In order for the approximator to work properly, the point set must be fairly dense, which in turn allows maximum flexibility in choosing the knots.

Although approximating data by B-spline curves to a given tolerance is an important problem in itself, the method forms the core of surface approximation to a user-specified accuracy. Because of the tensor product nature of B-sline surfaces, curve approximators facilitate u- and v-directional fittings, to which the surface approximation can be broken down.

5 Surface approximation

The curve approximation scheme can be used to approximate the point set

$$Q_{i,j} \quad i=0,\ldots,n \quad j=0,\ldots,m_i$$

up to the given tolerance ϵ. The idea is

1. Allocate percentages of ϵ to u- and v-directional approximations and to knot removal.
2. Compute an initial knot vector.
3. Pass this knot vector to each curve approximation.
4. Update the knot vector after each approximation.

The data approximation algorithm is explained as:

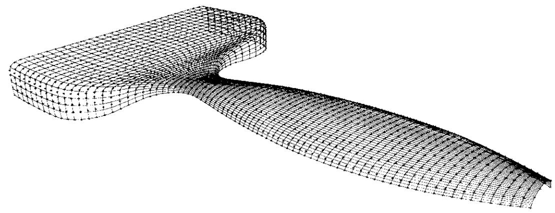
INPUT:
$Q_{i,j}$, $i=0,\ldots,n$; $j=0,\ldots,m_i$, data points
(p,q), required degrees
ϵ, tolerance
per_u, per_v, per_k, percentages
OUTPUT:
$S(u,v)$, $(P_{i,j}, U, V)$, surface approximant
ALGORITHM:
$\epsilon_u = per_u * \epsilon$;
$\epsilon_v = per_v * \epsilon$;
$\epsilon_k = per_k * \epsilon$;

$\hat{V} \leftarrow$ NULL;
for $i=0$ **to** n **do**
$C_i(v) \leftarrow$ approximate $Q_{i,j}$, $j=0,\ldots,m_i$, using $\hat{V}$ and ϵ_v;
$\hat{V} \leftarrow$ add new knots;
end
$\{V, \hat{m}\} \leftarrow$ make $C_i(v)$, $i=0,\ldots,n$, compatible;

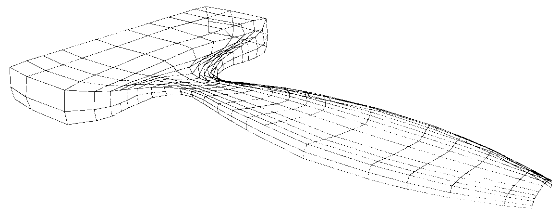
$\hat{U} \leftarrow$ NULL;
for $j=0$ **to** $\hat{m}$ **do**
for $i=0$ **to** n **do**
$R_i \leftarrow$ j-th control point of $C_i(v)$;
end
$C_j(u) \leftarrow$ approximate R_i, $i=0,\ldots,n$, using $\hat{U}$ and ϵ_u;
$\hat{U} \leftarrow$ add new knots;
end
$\{U, \hat{n}\} \leftarrow$ make $C_j(u)$, $i=0,\ldots,n$, compatible;

$\{P_{i,j}, i=0,\ldots,\hat{n}; j=0,\ldots,\hat{m}\} \leftarrow$ control points of $C_j(u)$, $j=0,..,\hat{m}$;
$S(u,v) \leftarrow$ remove all knots using ϵ_k;

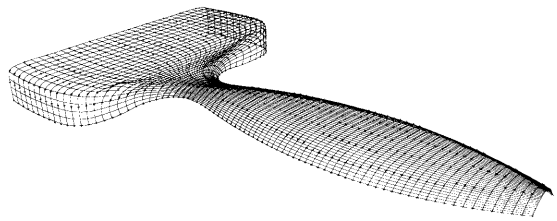
The routine first allocates tolerances to the three operators: u- and v-directional approximations and knot removal. Then the first knot vector in the v direction is initialized to null, which will then be computed inside the first call to the curve approximation routine. Each row of data points is then approximated up to ϵ_v, producing the v-directional curves $C_i(v)$, $i=0,\ldots,n$. Making these curves compatible, the highest index in the v direction and the v-knot vector are obtained. Performing the same procedure on the control points of $C_i(v)$ up to ϵ_u, one obtains the surface control points and the u-knot vector. The final surface is the result of knot removal using the tolerance ϵ_k.



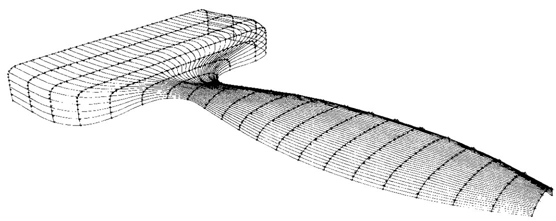
6a



6b



7a



7b

Fig. 6. a Surface approximation to 1554 points; **b** control points of surface approximant

Fig. 7a, b. Surface approximation to: **a** sparse points along dense rows; **b** dense points along sparse rows

The importance of allocating various fractions of ϵ to the three operators is illustrated in the examples of Figs. 6 and 7. In Fig. 6a, 81 rows of data points were approximated. The lowest row index is 6, the

Table 1. Results of tolerance allocation in examples in Figs. 6 and 7

per_u (percentage)	per_v (percentage)	per_k (percentage)	$\hat{n}$	$\hat{m}$	Total
33	33	33	25	16	442
20	60	20	57	14	870
60	20	20	18	19	380
50	50	0	22	14	345
0	100	0	104	13	1470
100	0	0	15	39	640
0	0	100	21	21	484

Table 2. Results of the approximating data points

ϵ	$\hat{n}$	$\hat{m}$
1.0	8	6
0.5	14	10
0.1	25	19
0.05	31	23
10^{-2}	46	33
10^{-3}	46	37
10^{-4}	48	38
10^{-5}	48	39
0.0	48	39

highest is 39, and the total number of points is 1554. The results of the tolerance allocation is shown in Table 1. The surface obtained by $(per_u, per_v, per_k) = (50\%, 50\%, 0\%)$ is depicted in Fig. 6a. The control points are shown in Fig. 6b. It is quite evident that the number of control points (as well as the quality of the surface) depends on how the tolerance is allocated. Our experience is summarized here.

- In general, it is not worth allocating too much of ϵ to knot removal because not too many knots remain to be removed after the approximations. A final knot-removal step is used only to eliminate knots that are too close together.
- The approximation strategy *interpolation-knot removal* is inferior to the method just mentioned, both in terms of number of control points and surface quality. Compare the fourth row (where only approximations are used) and the last row (where only interpolation and knot removal are employed) of results in Table 1.
- The allocation of ϵ depends on the data. If the rows are densely spaced, however, the points in each row are fairly sparse, a larger percentage is allocated to the u-directional approximation. An example is shown in Fig. 7a: 635 points are approximated with $(per_u, per_v, per_k) = (80\%, 20\%, 0\%)$. If the rows are sparsely spaced, however, each row contains a dense point set, and a higher percentage of ϵ is allocated to the v-directional approximation. Figure 7b shows such a situation: 526 points are approximated by $(per_u, per_v, per_k) = (20\%, 80\%, 0\%)$.

One may wonder how the number of control points depends on the tolerance, i.e., what kind of growth

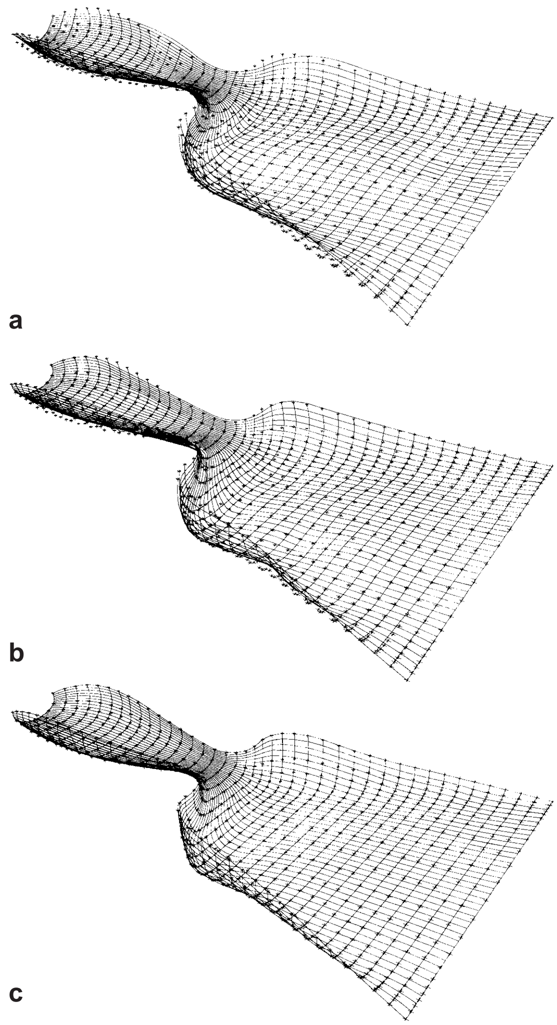


Fig. 8a–c. Surface approximation examples: **a** $\epsilon = 1.0$; **b** $\epsilon = 0.5$; **c** $\epsilon = 0.1$

rate the control points exhibit as the tolerance approaches zero. Table 2 shows the results for the data points in Fig. 8a–c. The relevant numerical values are: $n = 40$ and $(per_u, per_v, per_k) = (40\%, 40\%, 20\%)$.
The results of the first three rows are illustrated in Fig. 8a–c. Because the rows of data points are approximated independently of one another, an exponential growth of control points is expected (precisely as in surface lofting). Fortunately, this is not the case, and is due to the knot-vector control. By the time the approximator approximated some number of rows of data points, the updated knot vector, which is passed on to every curve approximator, has enough knots that most of them can be used for approximation, and update is hardly ever necessary. Traditional lofting methods would require an order of magnitude more control points in both directions, especially for smaller tolerances.

6 Conclusions

We have presented a method to approximate rows of data points with a $C^{(p-1,q-1)}$ degree (p, q) B-spline surface. The main ideas are: (1) approximate the rows of data points independently of one another, (2) select the knots from a given knot vector, and (3) control the percentage of the tolerance allocated to curve approximations and knot removal. The independent approximation of each row eliminated wiggles in the surface shape and allowed a good parametrization because of the flexibility inherent in the passed in knot vector. This knot vector in turn produced a very slow growth (logarithmiclike) of control points as the function of the tolerance. Practical experience shows that there is significant flexibility in how the knots are selected and these selections have little effect on the resultant shape.

Acknowledgements. The work reported in this paper was supported by the National Science Foundation under grants No. DDI-9526119 and No. CDA-9724422 awarded to the University of South Florida.

References

1. Amenta N, Bern M, Kamvysselis M (1998) A new Voronoi-based surface reconstruction algorithm. SIGGRAPH ’98 Conference Proceedings. pp 415–421
2. Bajaj CL, Bernardini F, Xu G (1995) Automatic reconstruction of surfaces and scalar fields from 3-D scans. SIGGRAPH ’95 Conference Proceedings. pp 109–118
3. Cox MG (1971) Curve fitting with piecewise polynomials. J Inst Math Appl 8:36–52
4. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. SIGGRAPH ’96 Conference Proceedings. pp 303–312
5. Eck M, Hoppe H (1996) Automatic reconstruction of B-spline surfaces of arbitrary topological type. SIGGRAPH ’96 Conference Proceedings. pp 325–334
6. Guo B (1997) Surface reconstruction: from points to splines. Comput Aided Des 29:269–277
7. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle, W (1992) Surface reconstruction from unorganized points. Comput Graph 26:71–78
8. Hoppe H, DeRose T, Duchamp T, Halstead M, Jin H, McDonald J, Schweitzer J, Stuetzle W (1994) Piecewise smooth surface reconstruction. SIGGRAPH ’94 Conference Proceedings. pp 295–302
9. Lyche T, Morken K (1987) Knot removal for parametric B-spline curves and surfaces. Comput Aided Geom Des 4:217–230
10. Ma W (1994) NURBS-based CAD modeling from measured points of physical models. PhD Dissertation, Catholic University of Leuven, Leuven, Belgium
11. Ma W, He P (1998) B-spline surface local updating with unorganized points. Comput Aided Des 30:853–862
12. Ma W, Kruth J-P (1994) Mathematical modeling of free-form curves and surfaces from discrete points with NURBS. In: Laurent PJ, Le Méhauté A, Schumaker LL (eds) Curves and surfaces in geometric design. Peters AK, Wellesley, Mass., pp 319–326
13. Ma W, Kruth J-P (1995a) NURBS curve and surface fitting and interpolation. In: Daehlen M, Lyche T, Schumaker LL (eds) Mathematical methods for curves and surfaces. Vanderbilt University Press, Nashville, Tenn, pp 315–322
14. Ma W, Kruth J-P (1995b) Parametrization of randomly measured points for least-squares fitting of B-spline curves and surfaces. Comput Aided Des 27:663–675
15. Milroy MJ, Bradley C, Vickers GW, Weir DJ (1995) G^1 continuity of B-spline surface patches in reverse engineering. Comput Aided Des 27:471–478
16. Piegl L, Tiller W (1997) The $\mathcal{NURBS}$ Book, 2nd edn., Springer, New York, NY

17. Pratt MJ (1985) Smooth parametric surface approximation to discrete data. Comput Aided Geom Des 2:165–171
18. Puntambekar NV, Jablokow AG, Sommer HJ (1994) Unified review of 3-D model generation for reverse engineering. Comput Integrated Manufacturing Syst 7:259–268
19. Sarkar B, Menq CH (1991) Smooth-surface approximation and reverse engineering. Comput Aided Des 23:623–628
20. Sapidis NS, Besl PJ (1995) Direct construction of polynomial surfaces from dense range images through region growing. ACM Trans Graph 14:171–200
21. Schmitt FM, Barsky BA, Du WH (1986) An adaptive subdivision method for surface-fitting from sampled data. Comput Graph 20:179–188
22. Schoenberg IJ, Whitney A (1953) On Polya frequency functions III: the positivity of translation determinants with an application to the interpolation problem by spline curves. Trans Am Math Soc 74:246–259
23. Tuohy ST, Maekawa T, Shen G, Patrikalakis NM (1997) Approximation of measured data with interval B-splines. Comput Aided Des 29:791–799
24. Várady T, Martin RR, Cox J (1997) Reverse engineering of geometric models – an introduction. Comput Aided Des 29:255–268

LES A. PIEGL is professor in the Department of Computer Science and Engineering, University of South Florida, Tampa, Florida, USA. His research interests are in CAD/CAM, geometric modeling, data structures and algorithms, computer graphics and software engineering. He spent many years researching and implementing NURBS in academia as well as in industry. He is the co-author of the textbook *The NURBS Book*, published by Springer-Verlag. He serves as Editor for *Computer-Aided Design*.

WAYNE TILLER is president ofGeomWare, Inc., a company specializing in NURBS technology and software. He has 28 years experience in applied mathematics, computer science, and software development. He has worked in the area of NURBS geometry since 1981, conducting research and implementing software. He has published numerous papers on this topic and is co-author of the book, *The NURBS Book*. He received a PhD in mathematics from Texas Christian University, USA.