Quasi-interpolants Based Multilevel B-Spline Surface Reconstruction from Scattered Data*

Byung-Gook Lee[1], Joon-Jae Lee[1] and Ki-Ryoung Kwon[2]

[1] Division of Internet Engineering, Dongseo University, Busan 617-716, Korea
lbg, jjlee@dongseo.ac.kr
[2] Division of Computer and Electronic Engineering, Pusan University of Foreign Studies, Busan 608-738, Korea
krkwon@pufs.ac.kr

Abstract. This paper presents a new fast and local method of 3D surface reconstruction for scattered data. The algorithm makes use of quasi-interpolants to compute the control points from a coarse to fine hierarchy to generate a sequence of bicubic B-spline functions whose sum approaches to the desired interpolation function. Quasi-interpolants gives a procedure for deriving local spline approximation methods where a B-spline coefficient only depends on data points taken from the neighborhood of the support corresponding B-spline. Experimental results demonstrate that high-fidelity reconstruction is possible from a selected set of irregular samples.

1 Introduction

The problem of recovering a surface from scattered data is one of those interesting problems that is simple in concept but tricky when get into the detail. As we know, the real world is made up of continuous surfaces, not discrete points. So, we want to create a continuous surface from the unorganized data points. The ultimate goal of this paper is a surface reconstruction method as getting a smooth and high fidelity of 3D surface from scattered data points. In particular, the description should be sufficiently completed to reconstruct the 3D surface within a certain tolerance error, given their relative locations and expected noise.

There exist many techniques for surface approximation to improve the approximate continuity and smoothness in handling scattered data[1,6,7,8]. Tensor product of B-splines surfaces is widely used to approximate rather than to work with other types of approximation because of the advantages inherent in working with tensor products. Tensor product guarantee internal continuity if the knot vectors are set properly.

This paper is based on the multilevel B-splines approximation techniques presented by the publication of Lee, Wolberg and Shin[11]. In the previous works,

* This work was supported by grant No.R01-2004-000-10851-0, R05-2004-000-10968-0 from Ministry of Science & Technology.

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3482, pp. 1209–1218, 2005.

Forsey and Bartels[5] developed a surface fitting method which is adaptive on hierarchical spline functions. However, this method cannot deal with scattered data. Lee presented a multilevel B-spline algorithm to fit a uniform bicubic B-spline surface to scatterd data where multilevel or hierarchy is used to reduce the approximation errors. Although the previous methods are processed locally, they can not only be computationally expensive when they applied to the large number of points sets, but also not guarantee a reasonable global approximation at initial level.

The splines approximation technique used in this paper is quasi-interpolants, first developed by de Boor and Fix[3]. The quasi-interpolants operators were later generalized by Lyche and Schumaker[9], and it is their version that used in the alternative surface approximation technique. A quasi-interpolants operator approximates a curve by calculating coefficients that are used to weight samplings of the curve to be approximated. The Lyche and Schumaker quasi-interpolants operator uses coefficients that are inexpensive to calculate and samplings that are relatively expensive to calculate. It turns out to produce splines approximation with the required accuracy.

We introduce a new algorithm using quasi-interpolants to implement the multilevel B-spline approximation and apply to scattered data. The proposed method converges in a few iterates while maintaining the accuracy. This algorithm achieved C^2-continuous interpolation function from arbitrary scattered data with numerically stable. The algorithm is described in section 2. Section 3 gives the explanation on how to reconstruct the quasi-interpolants. Then, section 4 shows the experimental results for numerical examples and finally, conclusions are given in section 5.

2 Multilevel B-Spline Approximation

The methods explored in this paper take a set of scattered data as input and produce tensor product B-spline surfaces as output. The algorithms run in a multiresolutional setting over uniform partitions such that the final surface f is composed of a sequence of surfaces at dyadic scales,

$$f = f_0 + f_1 + \ldots + f_k,$$

where $f_i \in S_i, i = 0, 1, \ldots, k$, and $S_0, S_1, \ldots, S_k$ is a nested sequence of subspaces of S_k,

$$S_0 \subset S_1 \subset \ldots \subset S_k.$$

The basic algorithms used for the results presented in this paper were published in 1997 by Lee, Wolberg and Shin. They called the schemes *Multilevel B-splines.* Our interest is mainly scattered data interpolation and approximation, which is also the main focus in [11].

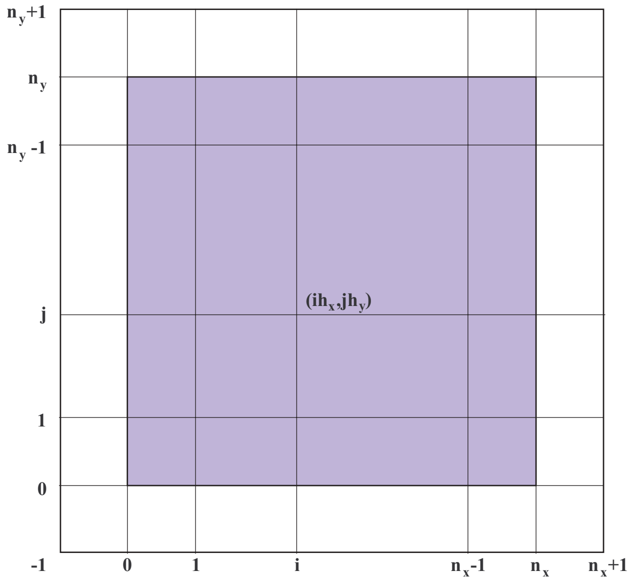

Fig. 1. The configuration of control lattice Ω

2.1 The Basic Schemes of B-Spline Approximation

Given a set of scattered points $P = \{P_i\}_{i=1}^{n}$, $P_i = (x_i, y_i, z_i) \in R^3$ and let $\Omega = \{(x, y)|0 \leq x < m_x, 0 \leq y < m_y\}$ be a rectangular domain in the xy-plane such that (x_i, y_i) is a point in Ω. Let Φ be a control lattice overlaid on a domain Ω. The control lattice Φ is an uniform tensor product grids over Ω.

To approximate scattered data points P, we formulate initial approximation function f as a uniform bicubic B-spline function, which is defined by a control lattice Φ. Let the initial number of control points on the lattice as $n_x = m_x/h_x$ in x-axis, and $n_y = m_y/h_y$ in y-axis. The knot intervals are uniform interval defined as h_x in x-axis and h_y in y-axis. So, for uniform cubic B-spline case, degree $d = 3$ and the set of knot vectors are defined as below:

$$\tau_x = \{-dh_x, \ldots, 0, h_x, \ldots, n_x h_x, \ldots, (n_x + d)h_x\}$$
$$\tau_y = \{-dh_y, \ldots, 0, h_y, \ldots, n_y h_y, \ldots, (n_y + d)h_y\}.$$

Let c_{ij} be the value of the ij-th control point on lattice Φ, located at position (ih_x, jh_y) of the grid defined by Φ, for $i = -1, 0, 1, \ldots, n_x + 1$ and $j = -1, 0, 1, \ldots, n_y + 1$. The approximation function f is defined in terms of these control points at position $(x, y) \in \Omega$ is given as

$$f(x, y) = \sum_{i=1}^{n_x+1} \sum_{j=1}^{n_y+1} c_{ij} B_{i,d}(x) B_{j,d}(y) \tag{1}$$

where $B_{i,d}$ and $B_{j,d}$ are uniform cubic B-spline basis functions, $d = 3$ and knot vector for cubic B-spline basis are below:

$$\{(i-2)h_x, (i-1)h_x, ih_x, (i+1)h_x, (i+2)h_x\}$$
$$\{(j-2)h_y, (j-1)h_y, jh_y, (j+1)h_y, (j+2)h_y\}.$$

2.2 Multilevel B-Spline Approximation

B-spline approximation(BA) algorithm generates a tradeoff exiting between the shape smoothness and accuracy of the approximation function. To overcome this tradeoff, multilevel B-splines approximation(MBA) algorithm is introduced [11]. The algorithm makes use of a hierarchy of control lattices to generate a sequence of function f_i and the final approximation function f is defined as the sum of functions f_i,

$$f = \sum_{i=1}^{k} f_i. \tag{2}$$

To optimize this process, B-spline refinement is used to reduce the sum of these functions into one equivalent B-spline function. The MBA algorithm serves result as smooth initial approximation f_0 to $\Delta^0 P = P$ defined on the coarsest control lattice $\Phi_0 = \Phi$, by applying the BA algorithm. To continue to the finer levels, below explanation are quoted from [11]: The first approximation possibly leaves large discrepancies at the data points in P. In particular, f_0 leaves a deviation

$$\Delta^1 z_i = z_i - f_0(x_i, y_i) \text{ for } i = 0, \ldots, n. \tag{3}$$

The next finer control lattice Φ_1 is then used to obtain function f_1 that approximates the difference $\Delta^1 P = \{(x_i, y_i, \Delta^1 z_i)\}$.

Then, the sum of $f_0 + f_1$ yields a smaller deviation (3) for each (x_i, y_i) in Ω.

$$\Delta^2 z_i = z_i - f_0(x_i, y_i) - f_1(x_i, y_i) \text{ for } i = 0, \ldots, n.$$

In general, for each level k in the hierarchy, the point set $\Delta^k P = \{(x_i, y_i, \Delta^k z_i)\}$ is approximated by a function f_k defined over the control lattices Φ_k, where

$$\Delta^k z_i = z_i - \sum_{l=0}^{k-1} f_l(x_i, y_i) = \Delta^{k-1} z_i - f_{k-1}(x_i, y_i)$$

and $\Delta^0 z_i = z_i$. This process starts from the coarsest lattice Φ_0 and continue incrementally to the finest lattice Φ_k with the set of knot vectors are defined as below:

$$\tau_x^k = \{-d\frac{h_x}{2^k}, \ldots, 0, \frac{h_x}{2^k}, \ldots, 2^k n_x \frac{h_x}{2^k}, \ldots, (2^k n_x + d)\frac{h_x}{2^k}\}$$
$$\tau_y^k = \{-d\frac{h_y}{2^k}, \ldots, 0, \frac{h_y}{2^k}, \ldots, 2^k n_y \frac{h_y}{2^k}, \ldots, (2^k n_y + d)\frac{h_y}{2^k}\}.$$

The final approximation function f is defined as the sum of the functions (2). They are many methods for refining a control lattice into another so that they generate the same B-spline functions. In this paper, B-spline refinement of an $(n_x + 3) \times (n_y + 3)$ control lattice $\Phi_0 = \Phi$ is always refined to a $(2^k n_x + 3) \times (2^k n_y + 3)$ control lattice Φ_k whose the control point spacing is half.

3 Quasi-interpolants

Many applications of splines make use of some approximation method to produce a spline function from given discrete data. Popular methods include interpolation and least squares approximation. However, both of these methods require solution of a linear system of equations with as many unknowns as the dimension of the spline space, and are therefore not suitable for real-time processing of large streams of data. For this purpose local methods, which determine spline coefficients by using only local information, are more suitable. To ensure good approximation properties it is important that the methods reproduce polynomials and preferably the functions in the given spline space. A method based on derivative information was constructed in [3], while a more general class was studied in [9]. In order to reproduce the spline space, the local information of the methods in [9] was restricted to lie in one knot interval. In this paper we remove this restriction. We then discuss some specific approximation methods for quadratic and cubic splines. We use B-splines as a basis for splines and denote the i^{th} B-spline of degree d with knots τ by $B_{i,d} = B_{i,d,\tau}$, and the linear space spanned by these B-splines by $S_{d,\tau}$.

Given a function f, the basic problem of spline approximation is to determine B-spline coefficients $(c_i)_{i=1}^n$ such that

$$Pf = \sum_{i=1}^{n} c_i B_{i,d}$$

is a reasonable approximation to f. The basic challenge is therefore to devise a procedure for determining the B-spline coefficients. We assume that f is defined on an interval $[a, b]$, and that we have selected a space of splines $S_{d,\tau}$ defined on $[a, b]$ (i.e., so that $\tau = (t_j)_{j=1}^{n+d+1}$ is nondecreasing with $t_{d+1} = a$ and $t_{n+1} = b$).

When determining c_k, this procedure gives us the freedom to restrict our attention to a local subinterval $I = [t_\mu, t_\nu]$ of our choice. By doing this we may reduce the complexity of the problem. Secondly, we have the freedom to choose the local approximation method P_I. Typical choices will be interpolation, least squares approximation, or a smoothing spline.

A general class of approximation methods are obtained by letting P_I be given as point functionals of the form

$$\lambda_{k,j} f = f(x_{k,j}) \text{ for } j = 1, \ldots, m_k,$$

where $m_k = \nu - \mu + d$ and $x_{k,1}, \ldots, x_{k,m_k}$ are given points. With this choice, it is well known (see page 200 of [1]) that if

$$B_{\mu-d-1+j,d}(x_{k,j}) > 0 \text{ for } j = 1, \ldots, m_k,$$

then we obtain c_k in the form

$$c_k = \lambda_k f = \sum_{j=1}^{m_k} w_{k,j} f(x_{k,j}), \tag{4}$$

for some vector $w_k = (w_{k,j})$. Equivalently, we can find w_k by solving the linear system

$$\delta_{i,k} = \lambda_k(B_{i,d}) = \sum_{j=1}^{m_k} w_{k,j} B_{i,d}(x_{k,j}), \tag{5}$$

for $i = \mu - d, \ldots, \nu - 1$ where $\delta_{i,k} = 1$ if $i = k$ and $\delta_{i,k} = 0$ otherwise, as usual. In practice one would usually determine c_k numerically, either from (4), or (5), except in special cases where the formulas are particularly simple. We consider some examples in the case where the knots and the degree of the spline are given.

Example 1. In the cubic spline case ($d = 3$). To determine coefficient c_k, we choose the interval $I = [t_k, t_{k+4}]$ which means that the local spline space has dimension 7,

$$S_{d,\tau,I} = \text{span}\{B_{k-3,d}, B_{k-2,d}, \ldots, B_{k+3,d}\}.$$

Here, the data points $\{P_{k,i}\}_{i=1}^{m_k}, P_{k,i} = (x_{k,i}, y_{k,i}) \in R^2$ are restricted to lie in the interval $I = [t_k, t_{k+4}]$.

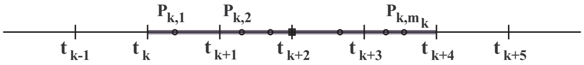

Fig. 2. A cubic spline, $I = [t_\mu, t_\nu] = [t_k, t_{k+4}]$

Coefficient matrix :

$$\begin{bmatrix} B_{k-3,3}(x_{k,1}) & B_{k-3,3}(x_{k,2}) & \ldots & B_{k-3,3}(x_{k,m_k}) \\ B_{k-2,3}(x_{k,1}) & B_{k-2,3}(x_{k,2}) & \ldots & B_{k-2,3}(x_{k,m_k}) \\ \vdots & \vdots & \ddots & \vdots \\ B_{k+3,3}(x_{k,1}) & B_{k+3,3}(x_{k,2}) & \ldots & B_{k+3,3}(x_{k,m_k}) \end{bmatrix}$$

The tensor product of the two spline spaces is defined as a family of all functions of the form

$$(Pf)(x, y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} c_{ij} B_{i,d}(x) B_{j,d}(y)$$

where $B_{i,d}$ and $B_{j,d}$ are the B-splines on $\tau_x = (t_j)_{j=1}^{n_x+d+1}$ and $\tau_y = (s_j)_{j=1}^{n_y+d+1}$ respectively.

Example 2. In the tensor product cubic spline case ($d = 3$). To determine coefficient c_{ij}, we choose the interval $I = [t_\mu, t_\nu] \times [s_\mu, s_\nu] = [t_i, t_{i+4}] \times [s_j, s_{j+4}]$ which means that the local spline space has dimension 49,

$$S_{d,\tau,I} = \text{span}\{B_{i-3,d}B_{j-3,d}, \dots, B_{i+3,d}B_{j+3,d}\}.$$

Here, the data points $\{P_{ij,k}\}_{k=1}^{m_{ij}}, P_{ij,k} = (x_{ij,k}, y_{ij,k}, z_{ij,k}) \in R^3$ are restricted to lie in the interval $I = [t_i, t_{i+4}] \times [s_j, s_{j+4}]$

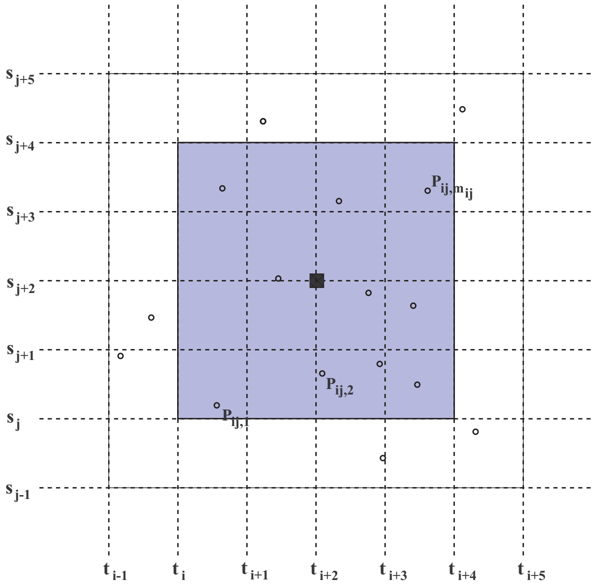

Fig. 3. A tensor product cubic spline, $I = [t_\mu, t_\nu] \times [s_\mu, s_\nu] = [t_i, t_{i+4}] \times [s_j, s_{j+4}]$

Coefficient matrix :

$$\begin{bmatrix} B_{i-3,3}(x_{ij,1})B_{j-3,3}(y_{ij,1}) & \dots & B_{i-3,3}(x_{ij,m_{ij}})B_{j-3,3}(y_{ij,m_{ij}}) \\ B_{i-3,3}(x_{ij,1})B_{j-2,3}(y_{ij,1}) & \dots & B_{i-3,3}(x_{ij,m_{ij}})B_{j-2,3}(y_{ij,m_{ij}}) \\ \vdots & \ddots & \vdots \\ B_{i+3,3}(x_{ij,1})B_{j+3,3}(y_{ij,1}) & \dots & B_{i+3,3}(x_{ij,m_{ij}})B_{j+3,3}(y_{ij,m_{ij}}) \end{bmatrix}$$

4 Experimental Results

To demonstrate the accuracy of reconstruction by the proposed algorithm, we performed experiments with the same test functions used in [11]. Given a test function $g(x, y)$, we first sampled data points from it and applied the algorithm to obtain an approximation function f. The difference between g and f is then measured by computing the normalized RMS(root mean square) error which is divided the RMS error by the difference of maximum and minimum values of g between the function values on a dense grid. That is,

$$RMS = \sqrt{\frac{\sum_{i=0}^{M}\sum_{j=0}^{N}(g(x_i, y_j) - f(x_i, y_j))^2}{(M+1)(N+1)}}$$

where $x_i = i/M$, $y_j = j/N$, and $M = N = 50$.

The test functions are

$$\begin{aligned}
g_1(x,y) &= 0.75\exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] \\
&+ 0.75\exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right] \\
&+ 0.5\exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] \\
&- 0.2\exp\left[-(9x-4)^2-(9y-1)^2\right] \\
g_2(x,y) &= (\tanh(9-9x-9y)+1)/9 \\
g_3(x,y) &= (1.25+\cos(5.4y))/(6+6(3x-1)^2) \\
g_4(x,y) &= \exp\left[-\frac{81}{4}((x-0.5)^2+(y-0.5)^2)\right]/3 \\
g_5(x,y) &= \sqrt{64-81((x-0.5)^2+(y-0.5)^2)}/9-0.5
\end{aligned}$$

where the domain is $\{(x,y)|0 \leq x \leq 1, 0 \leq y \leq 1\}$

For each test function, we used three data sets, M100, M500 and R500, where M100 and M500 are small and large data sets, which consist of 100 and 500

Table 1. Normalized RMS errors between test functions and their approximations

M100	g_1	g_2	g_3	g_4	g_5
1	.01451	.03020	.00693	.00215	.00025
2	.01174	.03345	.00515	.00830	.00024
3	.00902	.02800	.00447	.00540	.00022
M500	g_1	g_2	g_3	g_4	g_5
1	.01176	.02747	.00567	.00163	.00016
2	.00416	.00776	.00046	.00148	.00002
3	.00047	.00115	.00017	.00024	.00002
R500	g_1	g_2	g_3	g_4	g_5
1	.01328	.03098	.00627	.00189	.00021
2	.00689	.01367	.00079	.00166	.00006
3	.00244	.00801	.00034	.00053	.00005

points, respectively. We uniformly sampled 7 × 7 and 15 × 15 data points, respectively, while the others was randomly sampled. And R500 points were totally randomly sampled.

Fig. 4 shows 5 tested functions used in the experiment. Fig. 5 shows approximation surface of g_1 and error surface at initial level, respectively, where the circle represent the sampled data. The second level approximation result is obtained from sum of A) and B). Table 1 demonstrates that the proposed method reconstructs test functions very accurately regardless of type of the function within a few level. We started from the number of control points of 7x7 at initial level to three levels. Particularly, it generates good approximation corresponding to reasonable global approximation at initial level for smooth functional surface.

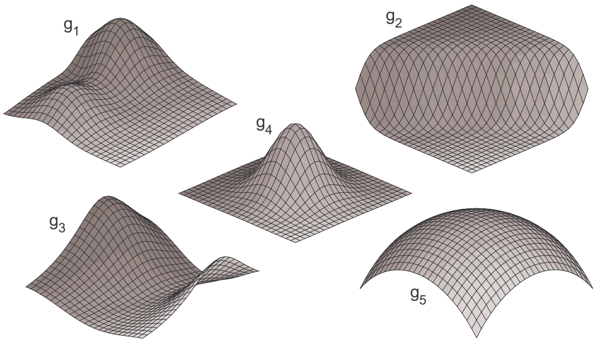

Fig. 4. Test functions

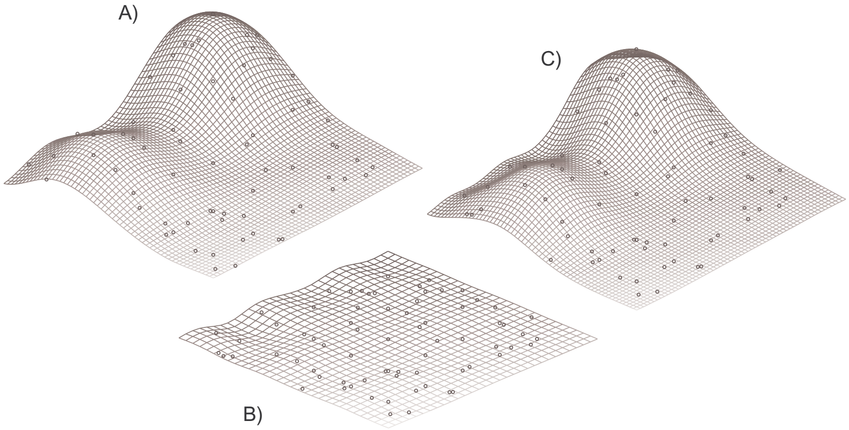

Fig. 5. A) Initial approximation surface from g_1 B) Error surface with $\Delta^1 z = z_i - f_0(x_i, y_i)$ C) Multilevel B-spline approximation with level 2, $f_0 + f_1$

5 Conclusion

This paper focuses on multilevel B-spline approximation based on quasi-interpolants for scattered data approximation and interpolation. The algorithm is fast and generates a C^2-continuous surface through a set of unevenly spaced points. Experimental results reveal that smooth 3D object reconstruction is possible from scattered data and irregular samples. Multilevel B-spline approximation was presented to circumvent the tradeoff which exists between the shape smoothness and approximation accuracy of the function, depending on the control lattice density. It is effectively gain in large performance. The quasi-interpolants is a special case of more general constructions and performs better approximation to reduces error results.

References

1. J. Barhak, A. Fischer, Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques, IEEE Trans. Visualization Comput. Graph. **7**(1) (2001) 1-16
2. de Boor, C., A Practical Guide to Splines, Springer Verlag, New York 1978
3. de Boor, C. and G.J.Fix, Spline approximation by quisiinterpolants, J. Approx. Theory **8** (1973) 19-45
4. Byung-Gook Lee, Lyche, T. and Knut Mørken, Some Examples of Quasi-Interpolants Constructed from Local Spline Projectors, Mathematical Methods for Curves and Surfaces, Oslo 2000, Lyche, T. and Schumaker L.L (eds), Vanderblit Press, Nashville (2001) 243-252
5. David Forsey and Richard Bartels. Surface fitting with hierarchical splines, ACM Trans. on Graphics **14** (1995) 134-161
6. P. Gu, X. Yan, Neural network approach to the reconstruction of free-form surfaces for reverse engineering, Comput. Aided Des. **27**(1) (1995) 59-64
7. M. Hoffmann, L. Varady, Free-form surfaces for scattered data by neural networks, J. Geometry Graph. **2** (1998) 1-6
8. A. Iglesias, G. Echevarria, A. Galvez, Functional networks for B-spline surface reconstruction, Future Generation Computer Systems **20** (2004) 1337-1353
9. Lyche, T. and L.L.Schumaker, Local spline approximation methods, J. Approx. Theory **25** (1979) 266-279
10. Øyvind Hjelle, Approximation of Scattered Data with Multilevel B-splines, Technical Report STF42 A01011, SINTEF 2001
11. S. Y. Lee, G. Wolberg, and S. Y. Shin. Scattered Data Interpolation with Multilevel B-Splines. IEEE Transactions on Visualization and Computer Graphics **3**(3) (1997) 229-244