Scattered Data Interpolation with Multilevel B-Splines

Seungyong Lee, George Wolberg, and Sung Yong Shin

Abstract—This paper describes a fast algorithm for scattered data interpolation and approximation. Multilevel B-splines are introduced to compute a C^2-continuous surface through a set of irregularly spaced points. The algorithm makes use of a coarse-to-fine hierarchy of control lattices to generate a sequence of bicubic B-spline functions whose sum approaches the desired interpolation function. Large performance gains are realized by using B-spline refinement to reduce the sum of these functions into one equivalent B-spline function. Experimental results demonstrate that high-fidelity reconstruction is possible from a selected set of sparse and irregular samples.

Index Terms—Scattered data interpolation, multilevel B-splines, data approximation.

1 Introduction

SCATTERED data interpolation refers to the problem of fitting a smooth surface through a scattered, or nonuniform, distribution of data samples. This subject is of practical importance in many science and engineering fields, where data is often measured or generated at sparse and irregular positions. The goal of interpolation is to reconstruct an underlying function (e.g., surface) that may be evaluated at any desired set of positions. This serves to smoothly propagate the information associated with the scattered data onto all positions in the domain.

There are three principal sources of scattered data: measured values of physical quantities, experimental results, and computational values [22]. They are found in diverse scientific and engineering applications. For example, nonuniform measurements of physical quantities are collected in geology, meteorology, oceanography, cartography, and mining; scattered experimental data is produced in chemistry, physics, and engineering; and nonuniformly spaced computational values arise in the output from finite element solutions of partial differential equations, and various applications in computer graphics and computer vision.

These fields require scattered data interpolation to determine values at arbitrary positions, not just those at which the data is available. This facilitates many useful operations for visualizing scattered multidimensional data. For instance, in medical imaging, scattered data interpolation is essential to construct a closed surface from CT or MRI images of human organs. In geological applications, the derived interpolation function facilitates a contour map to be plotted. Computer vision utilizes scattered data interpolation to perform visual surface reconstruction on sparse measurements obtained from feature-based stereo or motion. In image morphing, scattered data interpolation is useful for deriving a smooth mapping function from the correspondence of feature points between a pair of images. That same process is used to achieve image registration for remote sensing applications.

Despite a flurry of activity in this area, scattered data interpolation remains a difficult and computationally expensive problem. The vast literature devoted to this subject documents various approaches, many of which suffer from limitations in smoothness, time complexity, or allowable data distributions [22], [28]. This paper addresses these problems and introduces a very fast algorithm for constructing a C^2-continuous interpolation function from arbitrary scattered data. The algorithm applies an effective B-spline approximation technique to a hierarchy of control lattices to generate a sequence of functions whose sum approaches the desired interpolation function. Large performance gains are realized by using B-spline refinement to represent the sum of several functions as one B-spline function.

This paper is based on the multilevel B-spline approximation technique presented in [33], [34] for image morphing. In that work, multilevel B-splines were used to propagate user-specified values at scattered features across the image. This paper describes multilevel B-splines in terms of scattered data interpolation and significantly improves the performance by applying B-spline refinement to the control lattice hierarchy. In addition, we consider how multilevel B-spline approximation can be used to generate an interpolation function through scattered data points. We present a sufficient condition for interpolation and an adaptive representation of the control lattice hierarchy to minimize memory overhead.

Although our method applies to multivariate data, we limit our presentation to bivariate data for clarity. We assume that the independent data is 2D and the dependent

- *S. Lee is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, 790-784, Korea. E-mail: leesy@postech.ac.kr.*
- *G. Wolberg is with the Department of Computer Science, City College of New York, New York, NY 10031. E-mail: wolberg@cs-mail.engr.ccny.cuny.edu.*
- *S.Y. Shin is with the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, 305-701, Korea. E-mail: syshin@jupiter.kaist.ac.kr.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 105371.

data is a simple scalar. This is the familiar surface interpolation problem in which we have data values z_k defined at an arbitrary set of positions $\{(x_k, y_k)\}$. To interpolate the data, we seek to find a function $f(x, y)$ which takes on the value z_k at (x_k, y_k), i.e., $z_k = f(x_k, y_k)$. Together, (x_k, y_k, z_k) comprise the scattered bivariate data. The reader should note that other names for scattered data are offered in the literature, including "random," "nonuniform," "irregular," and "arbitrarily located."

This paper is organized as follows. Section 2 reviews previous work. B-Spline approximation is presented in Section 3. That serves to motivate the discussion of multilevel B-Spline approximation, given in Section 4. In Section 5, we demonstrate how the approximation algorithm is used to perform multilevel B-spline interpolation. Section 6 gives several examples in various applications. Performance results and a comparison to thin plate splines and hierarchical B-spline refinement are presented in Section 7. Conclusions are given in Section 8.

2 Previous Work

There is a vast amount of literature devoted to scattered data interpolation. The reader is referred to [22], [28], [44], [2] for excellent surveys. In this section, we review several dominant approaches based on Shepard's method, radial basis functions, thin plate splines, and finite element methods. We also consider related research in image processing and geometric modeling, including multiresolution filtering, direct manipulation techniques, and hierarchical B-spline refinement.

One of the earliest algorithms in this field was based on inverse distance weighting of data. Developed by meteorologists and geologists [9], [10], it has become known as Shepard's method [45]. Shepard defined a C^0-continuous interpolation function as the weighted average of the data, with the weights being inversely proportional to distance. This technique suffers from several shortcomings, including cusps, corners, and flat spots at the data points, as well as undue influence of points which are far away. Furthermore, it is a global method requiring all the weights to be recomputed if any data point is added, removed, or modified. Franke and Nielson introduced the modified quadratic Shepard's method [21] to address these deficiencies and produce C^1-continuous interpolation.

Another popular approach to scattered data interpolation is to define the interpolation function as a linear combination of radially symmetric basis functions, each centered at a data point. The unknown coefficients for the basis functions are determined by solving a linear system of equations. The coefficient matrix is always full, and, for large data sets, it may become poorly conditioned and require preconditioning [13], [12]. Popular choices for the basis functions include Gaussian, multiquadratics [27], and shifted log [19], [12], [22], [28]. Hardy's multiquadratics are among the most successful and applied methods due to its simplicity, well-conditioned property, and intuitive results. See [19], [20], [41] for numerous tests and comparisons. A survey of radial basis functions for image warping can be found in [43].

Thin plate splines are derived by minimizing the integral of the curvatures over the domain among the interpolation functions of the scattered points. They are widely used due to their visually pleasing results and stability for large data sets. Although they are usually formulated as the solution to a variational problem, Duchon has shown thin plate splines to be derived from radial basis functions [11]. The numerical solution of thin plate splines can be accelerated by the multigrid relaxation technique [4], [5], [47]. An alternative to multigrid relaxation has been presented that uses hierarchical basis functions or wavelets to accelerate the convergence of an iterative technique such as conjugate gradient descent [46], [25]. Nevertheless, the numerical solution remains computationally expensive when the interpolation function is computed on a large grid. Recently, thin plate splines have been used to generate smooth warp functions for image warping and morphing [35], [31], [32].

Another class of solutions to scattered data interpolation is due to finite element methods. This approach involves creating some type of optimal triangulation on the set of data points to delimit local neighborhoods over which surface patches are defined. These patches are constrained to interpolate the original data. There are several criteria suggested by Lawson [30] to derive optimal triangulations in which long thin triangles with small angles are avoided. Piecewise linear approximation over the triangulation is not smooth, achieving only C^0-continuity. The most common C^1 method uses the Clough-Tocher triangular interpolant [7], [2], [26]. A related technique was proposed in [39]. Triangulation methods, however, are sensitive to data distribution, i.e., long thin triangles cannot always be avoided.

Schumaker [44] proposed a two-stage method that first generates a grid of data using any method for scattered data interpolation. The second stage applies a standard tensor product approximation on the grid. The resulting approximation may be made to interpolate the original data through the use of an iterative process called the delta iteration, developed by Foley and Nielson [15]. Arge et al. [1] proposed an approximation scheme consisting of three steps: regularization, local approximation, and extrapolation. They first determine the approximation function by a local method at a subset of the grid points where the data density is high. That function is then extrapolated to the entire grid by a global method.

In the image processing community, scattered data interpolation is necessary to perform reconstruction among nonuniform samples. A fine survey of nonuniform reconstruction techniques can be found in [23]. A trend in recent algorithms has been the use of hierarchical, or multiresolution, filtering to extend onto all positions the information known only at the sparse and irregular samples. Burt proposed hierarchical polynomial fit filtering to yield a multiresolution set of low-pass filtered images that can be combined to form a smooth surface passing through the original data [6]. Mitchell proposed multistage filtering to handle highly variable sample density [40]. In that work, weighted-average filters are repeatedly applied with ever-narrowing low-pass cutoff until the proper bandwidth for the display is reached.

Recent work in geometric modeling has addressed scattered data interpolation. A B-spline approximation technique for scattered data was introduced to directly manipulate an object modeled by free-form deformation [29]. That technique calculates the pseudoinverse of a matrix containing B-spline basis function values to minimize the approximation error. Welch and Witkin proposed a variational approach to directly manipulate B-spline surfaces with scattered points or curves [50]. The control lattice is determined by minimizing the sum of energy functionals to derive a smooth interpolating surface. The high computational cost of pseudoinverse and energy minimization solutions render these methods prohibitively expensive for large data sets.

Forsey and Bartels proposed hierarchical B-spline refinement for object modeling [16]. They augmented B-spline approximation with that technique to interpolate a grid of data using a control lattice hierarchy [17], [18]. Interpolation is achieved by successively improving the approximation at a coarse level with a correction term from the next finer level. Although their method is similar in spirit to the multilevel B-spline approximation presented in this paper, their method cannot handle scattered data.

3 B-Spline Approximation

Recently, a B-spline approximation technique has been proposed for image morphing [33], [34]. In this section, we elaborate on that technique in terms of scattered data interpolation and present the details of the algorithm.

3.1 Basic Idea

Let $\Omega = \{(x, y) \mid 0 \le x < m,\ 0 \le y < n\}$ be a rectangular domain in the *xy*-plane. Consider a set of scattered points $P = \{(x_c, y_c, z_c)\}$ in 3D space, where (x_c, y_c) is a point in Ω. To approximate scattered data P, we formulate approximation function f as a uniform bicubic B-spline function, which is defined by a control lattice Φ overlaid on domain Ω. Without loss of generality, we assume that Φ is an $(m + 3) \times (n + 3)$ lattice which spans the integer grid in Ω (Fig. 1). Later, we shall consider the effect of different lattice sizes on the approximation function.

Let ϕ_{ij} be the value of the ij-th control point on lattice Φ, located at (i, j) for $i = -1, 0, ..., m + 1$ and $j = -1, 0, ..., n + 1$. The approximation function f is defined in terms of these control points by

$$f(x, y) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(s) B_l(t) \phi_{(i+k)(j+l)}, \tag{1}$$

where $i = \lfloor x \rfloor - 1$, $j = \lfloor y \rfloor - 1$, $s = x - \lfloor x \rfloor$, and $t = y - \lfloor y \rfloor$. B_k and B_l are uniform cubic B-spline basis functions defined as

$$B_0(t) = (1-t)^3 / 6,$$
$$B_1(t) = \left(3t^3 - 6t^2 + 4\right) / 6,$$
$$B_2(t) = \left(-3t^3 + 3t^2 + 3t + 1\right) / 6,$$
$$B_3(t) = t^3 / 6,$$

where $0 \le t < 1$. They serve to weigh the contribution of each control point to $f(x, y)$ based on its distance to (x, y).

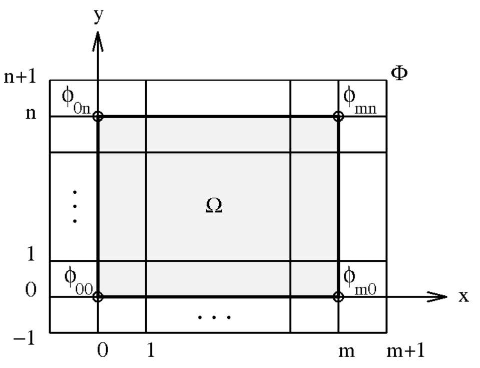


Fig. 1. The configuration of control lattice Ω.

With this formulation, the problem of deriving function f is reduced to solving for the control points in Φ that best approximate the scattered data in P.

To determine the unknown control lattice Φ, we first consider one data point (x_c, y_c, z_c) in P. From (1), we know that function value $f(x_c, y_c)$ relates to the sixteen control points in the neighborhood of (x_c, y_c). Without loss of generality, we may assume that $1 \le x_c, y_c < 2$. Then, control points ϕ_{kl}, for k, l = 0, 1, 2, 3, determine the value of f at (x_c, y_c). For function f to take on the value z_c at (x_c, y_c), the control points ϕ_{kl} must satisfy

$$z_c = \sum_{k=0}^{3} \sum_{l=0}^{3} w_{kl} \phi_{kl}, \tag{2}$$

where $w_{kl} = B_k(s) B_l(t)$ and $s = x_c - 1, t = y_c - 1$.

There are many values for the ϕ_{kl}'s that satisfy (2). We choose one in the least-squared sense that minimizes $\sum_{k=0}^{3} \sum_{k=0}^{3} \phi_{kl}^2$. This tends to minimize the deviation of f from zero over the domain Ω. This property will prove useful when we apply the B-spline approximation to a hierarchy of control lattices in Section 4. Simple linear algebra using pseudoinverse is used to derive the solution [29]:

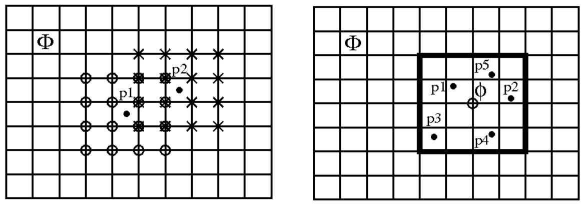
$$\phi_{kl} = \frac{w_{kl} z_c}{\sum_{a=0}^{3} \sum_{b=0}^{3} w_{ab}^2}. \tag{3}$$

In this solution, the control points near (x_c, y_c) get larger values because they are associated with larger w_{kl}. The resulting function f has the value z_c at (x_c, y_c) and tapers off smoothly.

Now, we consider all the data points in P. For each data point, (3) can be used to determine the set of 4×4 control points in its neighborhood. These neighborhoods may overlap for sufficiently close data points. Fig. 2a depicts two such data points, p_1 and p_2. They may assign different values to several shared control points. In general, we resolve multiple assignments to a control point ϕ by considering the data points in its 4×4 neighborhood (Fig. 2b). Only these points may influence the value of ϕ by (3). We call this set of points the *proximity data set* of ϕ.

Let P_{ij} be the proximity data set of control point ϕ_{ij} such that

$$P_{ij} = \left\{ (x_c, y_c, z_c) \in P \;\middle|\; i - 2 \le x_c < i + 2, j - 2 \le y_c < j + 2 \right\}.$$



(a) overlapping control points (b) proximity data set

Fig. 2. Positional relationship between data and control points.

For each point (x_c, y_c, z_c) in P_{ij}, (3) gives ϕ_{ij} a different value ϕ_c:

$$\phi_c = \frac{w_c z_c}{\sum_{a=0}^{3}\sum_{b=0}^{3} w_{ab}^2}, \tag{4}$$

where $w_c = w_{kl} = B_k(s)B_l(t), k = (i+1) - \lfloor x_c \rfloor, l = (j+1) - \lfloor y_c \rfloor$, $s = x_c - \lfloor x_c \rfloor$, $t = y_c - \lfloor y_c \rfloor$. To compromise among the values, ϕ_{ij} is chosen to minimize error $e(\phi_{ij}) = \sum_c (w_c\phi_{ij} - w_c\phi_c)^2$. The term $(w_c\phi_{ij} - w_c\phi_c)$ is the difference between the real and expected contributions of ϕ_{ij} to function f at (x_c, y_c). In other words, it is the approximation error due to ϕ_{ij}, assuming that the other control points surrounding (x_c, y_c, z_c) have their values determined by (3) using that data point. By differentiating the error $e(\phi_{ij})$ with respect to ϕ_{ij}, we get

$$\phi_{ij} = \frac{\sum_c w_c^2 \phi_c}{\sum_c w_c^2}. \tag{5}$$

The proximity data set P_{ij} is the set of points in P at which control point ϕ_{ij} has an influence on function f. When P_{ij} contains several points, (5) provides a least-squared solution to ϕ_{ij} which minimizes a local approximation error. When P_{ij} consists of only one point, (5) reduces to (3), leaving no approximation error. When P_{ij} is empty, however, ϕ_{ij} has no influence on $f(x_c, y_c)$ for any data point (x_c, y_c, z_c) in P. This implies that ϕ_{ij} may be assigned any arbitrary value, such as zero or the average of the z_c's, without affecting the approximation error. In that case, we assign zero to ϕ_{ij} to make function f tend to zero in its neighborhood. This will prove useful for multilevel B-spline approximation in Section 4.

3.2 Algorithm

To determine the control lattice Φ from the data points in P using (5), it is not necessary to explicitly identify the proximity data set for each control point. Since each data point in P influences a set of 4 × 4 neighboring control points in Φ, it belongs only to the proximity data sets of those control points. Hence, we can efficiently accumulate the numerator and denominator of (5) for each control point by considering each data point in turn. The value of a control point is then obtained by division if the denominator is not zero. A null denominator occurs only when a control point has an empty proximity data set. In that case, we assign zero to the control point. The following pseudocode outlines this B-spline approximation method, which we denote as the BA algorithm.

BA Algorithm

```
Input: scattered data $P = \{(x_c, y_c, z_c)\}$
Output: control lattice $\Phi = \{\phi_{ij}\}$
for all $i, j$ do let $\delta_{ij} = 0$ and $\omega_{ij} = 0$
for each point $(x_c, y_c, z_c)$ in $P$ do
    let $i = \lfloor x_c \rfloor - 1$ and $j = \lfloor y_c \rfloor - 1$
    let $s = x_c - \lfloor x_c \rfloor$ and $t = y_c - \lfloor y_c \rfloor$
    compute $w_{kl}$'s and $\sum_{a=0}^{3}\sum_{b=0}^{3} w_{ab}^2$
    for $k, l = 0, 1, 2, 3$ do
        compute $\phi_{kl}$ with (3)
        add $w_{kl}^2\phi_{kl}$ to $\delta_{(i+k)(j+l)}$
        add $w_{kl}^2$ to $\omega_{(i+k)(j+l)}$
    end
end
for all $i, j$ do
    if $\omega_{ij} \neq 0$ then
        compute $\phi_{ij} = \delta_{ij}/\omega_{ij}$
    else let $\phi_{ij} = 0$
end
```

The time and space complexity of the BA algorithm is $O(p + mn)$, where p is the number of data points, and $(m + 3) \times (n + 3)$ is the size of the control lattice. Although the control point values are determined locally, we minimize the approximation error so that the resulting function properly reflects the scattered data. Fig. 3 shows an example. Figs. 3a and 3b, respectively, show data P and the B-spline function f derived by the BA algorithm for $m = n = 16$. Notice that f nicely approximates P at densely distributed data points and interpolates P at isolated points.

The density of control lattice Φ overlaid on domain Ω directly affects the shape of approximation function f. If we select lattice Φ of size $(m' + 3) \times (n' + 3)$, instead of $(m + 3) \times (n + 3)$, then the ijth control point in the new lattice is located at $(i\frac{m}{m'}, j\frac{n}{n'})$ in Ω. As Φ becomes coarser, the proximity data set of each control point covers a larger number of points in P. This causes many data points to be blended together to yield a smoother shape for f at the expense of approximation accuracy. However, as Φ becomes finer, the influence of a data point is limited to smaller neighborhoods. This enables P to be more closely approximated, although f will tend to contain local peaks near the data points. These characteristics are evident in Figs. 3c and 3d, where $m = n = 8$ and $m = n = 32$, respectively.

The BA algorithm runs very fast even when the number of data points is large. Given 5,000 data points, for instance, the control lattice for the approximation function can be derived in 0.15 second on a Sun SPARC10. Furthermore, since B-splines have local support, only a small neighborhood in the control lattice needs to be updated when a data point is added or deleted. This modification is easily computed as long as we keep the variables δ_{ij} and ω_{ij}.

The approximation function f from the BA algorithm is C^2-continuous because it is a bicubic B-spline surface

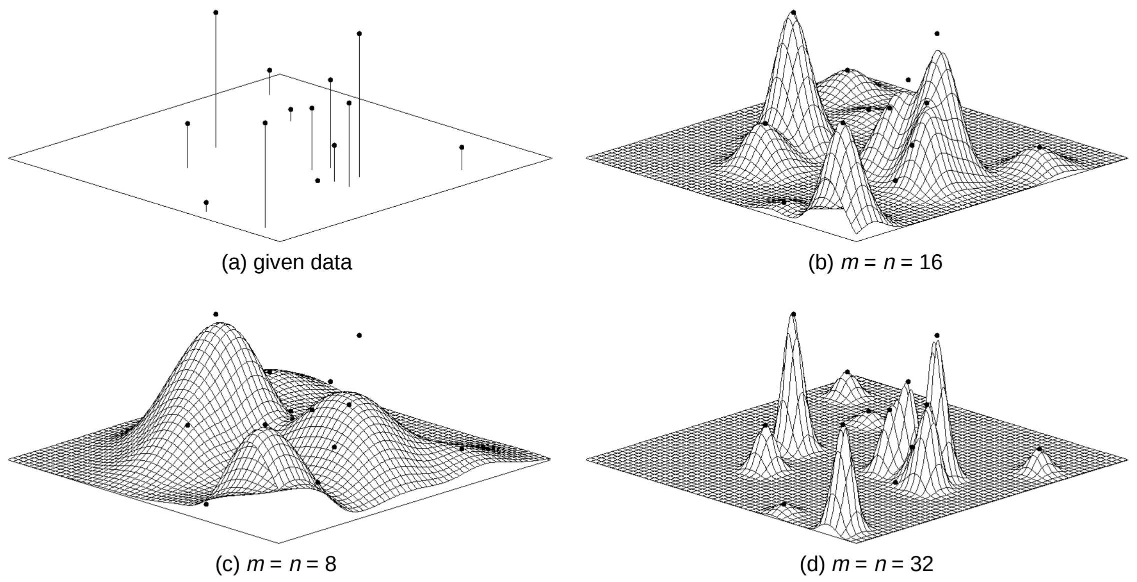


Fig. 3. Examples of B-spline approximation under different control lattice resolutions.

generated by control lattice Φ. Since Φ can be derived very quickly from the data points, most of the time for computing function f is taken by the evaluation of f from Φ on domain Ω. To accelerate this evaluation, we use the forward difference method [14], [51] or table lookup for basis functions depending on the control point spacing. It takes 0.77 second for a Sun SPARC10 to obtain function f on a $1{,}024 \times 1{,}024$ grid from the data points in Fig. 3.

4 Multilevel B-Spline Approximation

A tradeoff exists between the shape smoothness and accuracy of the approximation function generated by the BA algorithm. In this section, we present a multilevel B-spline approximation algorithm to circumvent this tradeoff. The resulting function simultaneously achieves a smooth shape while closely approximating the given data P. The algorithm makes use of a hierarchy of control lattices to generate a sequence of functions f_k whose sum approaches the desired approximation function. In the sequence, a function from a coarse lattice provides a rough approximation, which is further refined in accuracy by functions derived from finer lattices. We further optimize this process by using B-spline refinement to reduce the sum of these functions into one equivalent B-spline function.

4.1 Basic Algorithm

Consider a hierarchy of control lattices, $\Phi_0, \Phi_1, \ldots, \Phi_h$, overlaid on domain Ω. We assume that the spacing between control points for Φ_0 is given and that the spacing is halved from one lattice to the next. Therefore, if Φ_k is an $(m + 3) \times (n + 3)$ lattice, the next finer lattice Φ_{k+1} will have $(2m + 3) \times (2n + 3)$ control points. The position of the ij-th control point in Φ_k coincides with that of the $(2i, 2j)$-th control point in Φ_{k+1}.

The multilevel B-spline approximation begins by applying the BA algorithm to P with the coarsest control lattice Φ_0. The resulting function f_0 serves as a smooth initial approximation that possibly leaves large discrepancies at the data points in P. In particular, f_0 leaves a deviation $\Delta^1 z_c = z_c - f_0(x_c, y_c)$ for each point (x_c, y_c, z_c) in P. The next finer control lattice Φ_1 is then used to obtain function f_1 that approximates the difference $P_1 = \{(x_c, y_c, \Delta^1 z_c)\}$. Then, the sum $f_0 + f_1$ yields a smaller deviation $\Delta^2 z_c = z_c - f_0(x_c, y_c) - f_1(x_c, y_c)$ for each point (x_c, y_c, z_c) in P.

In general, for a level k in the hierarchy, we derive function f_k by using control lattice Φ_k to approximate data $P_k = \{(x_c, y_c, \Delta^k z_c)\}$, where $\Delta^k z_c = z_c - \sum_{i=0}^{k-1} f_i(x_c, y_c) = \Delta^{k-1} z_c - f_{k-1}(x_c, y_c)$, and $\Delta^0 z_c = z_c$. This process starts from the coarsest lattice Φ_0 and continues incrementally to the finest lattice Φ_h. The final approximation function f is defined as the sum of functions f_k, i.e., $f = \sum_{k=0}^{h} f_k$. Note that only the coarsest lattice Φ_0 is applied to the original data P to derive the global shape of function f. All successively finer lattices serve to approximate and remove the residual error. In this manner, we have an incremental solution for function f that yields a smooth and close approximation to P. The following pseudocode outlines the basic algorithm for multilevel B-spline approximation, which we denote as the basic MBA algorithm. Note that a hierarchy of control lattices is sufficient to represent function f because each f_k can be represented by Φ_k, and f is the sum of the f_k's.

```
**Basic MBA Algorithm**
Input: scattered data $P = \{(x_c, y_c, z_c)\}$
Output: a control lattice hierarchy $\Phi_0, \Phi_1, \ldots, \Phi_h$
let $k = 0$
while $k \le h$ do
    let $P_k = \{(x_c, y_c, \Delta^k z_c)\}$
    compute $\Phi_k$ from $P_k$ by the BA algorithm
    compute $\Delta^{k+1} z_c = \Delta^k z_c - f_k(x_c, y_c)$ for each data point
    let $k = k + 1$
end
```

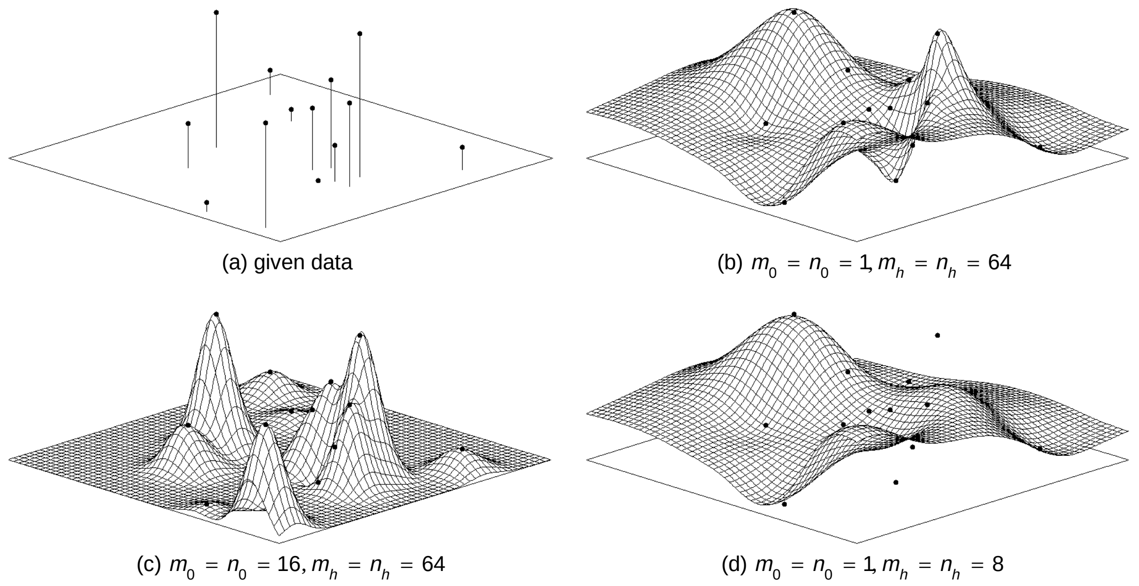


Fig. 4. Examples of multilevel B-spline approximation under different resolutions for the coarsest and finest control lattices.

Let p be the number of data points in P and let $(m + 3) \times (n + 3)$ be the size of the finest control lattice Φ_h. The number of control points in lattice Φ_k is a quarter of that in the next finer lattice Φ_{k+1}. Hence, the time complexity of the basic MBA algorithm is $O(p + mn) + O(p + \frac{1}{4}mn) + \ldots + O(p + \frac{1}{4^h}mn) = O(hp + \frac{4}{3}mn)$. The space complexity is $O(p + \frac{4}{3}mn)$ because we have to store all the control lattices in the hierarchy. A function from the basic MBA algorithm is C^2-continuous because it is the sum of C^2-continuous B-spline functions. Fig. 4b shows an example. The algorithm is applied to the data shown in Fig. 4a, which is the same as that in Fig. 3a. Notice that multilevel B-spline approximation generates a much smoother and more accurate function than B-spline approximation.

In the multilevel B-spline approximation, the density of the coarsest lattice Φ_0 determines the area of influence of a data point on function f. Larger spacing between control points serves to merge the influences of data points over large areas to produce smoother function shapes. On the other hand, the density of the finest lattice Φ_h controls the precision with which f approximates the data points. When Φ_h is sufficiently fine compared to the data distribution, f can interpolate the data without an approximation error.

Fig. 4 shows several examples. In Fig. 4a, the given data is the same as that in Fig. 3a. Let $(m_k + 3) \times (n_k + 3)$ be the size of control lattice Φ_k. In Fig. 4b, $m_0 = n_0 = 1$ and $m_h = n_h = 64$. In the figure, the approximation function has a smooth shape and interpolates the data points. Fig. 4c depicts the function when Φ_0 gets finer $m_0 = n_0 = 16$. In Fig. 4d, a coarser Φ_h with $m_h = n_h = 8$ results in an approximation error.

In practice, we use the same horizontal and vertical spacing of control points in a control lattice. The sizes of control lattices in the hierarchy are related to the aspect ratio of domain Ω. For the coarsest control lattice Φ_0, we generally choose $m_0 = 1$ or $n_0 = 1$, depending on the aspect ratio, to make Φ_0 as coarse as possible. Then, the basic MBA algorithm is applied through the hierarchy until we reach a sufficiently fine control lattice Φ_h for which the maximum error falls below a specified threshold. This results in an approximation function that satisfies our goal for smooth function shape and accuracy.

4.2 Optimization with B-Spline Refinement

The basic MBA algorithm generates a control lattice hierarchy that represents the approximation function f. To evaluate f, we must determine function f_k from control lattice Φ_k for each level k, and add them over domain Ω (Fig. 5a). This introduces a significant overhead in computation time if f has to be evaluated at a large number of points in Ω. We propose to address this problem by progressively applying B-spline refinement to the control lattice hierarchy. This allows f to be represented by one B-spline function rather than the sum of several B-spline functions. Consequently, the computation of f_k is limited to the small number of control points in Φ_k rather than all the points in Ω.

Let $F(\Phi)$ be the B-spline function generated by control lattice Φ and let $|\Phi|$ denote the size of Φ. With B-spline refinement, we can derive control lattice Φ_0' from the coarsest lattice Φ_0 such that $F(\Phi_0') = f_0$ and $|\Phi_0'| = |\Phi_1|$. Then, the sum of functions f_0 and f_1 can be represented by control lattice Ψ_1 which results from the addition of each corresponding pair of control points in Φ_0' and Φ_1. That is, $F(\Psi_1) = g_1 = f_0 + f_1$, where $\Psi_1 = \Phi_0' + \Phi_1$.

In general, let $g_k = \sum_{i=0}^{k} f_i$ be the partial sum of functions f_i up to level k in the hierarchy. Suppose that function g_{k-1} is represented by a control lattice Ψ_{k-1} such that $|\Psi_{k-1}| = |\Phi_{k-1}|$. In the same manner as we computed Ψ_1 above, we can refine Ψ_{k-1} to obtain Ψ_{k-1}', and add Ψ_{k-1}' to Φ_k to derive Ψ_k such that $F(\Psi_k) = g_k$ and $|\Psi_k| = |\Phi_k|$. That is, $\Psi_k = \Psi_{k-1}' + \Phi_k$. Therefore, from $g_0 = f_0$ and $\Psi_0 = \Phi_0$, we can compute a sequence of control lattices Ψ_k to progres-

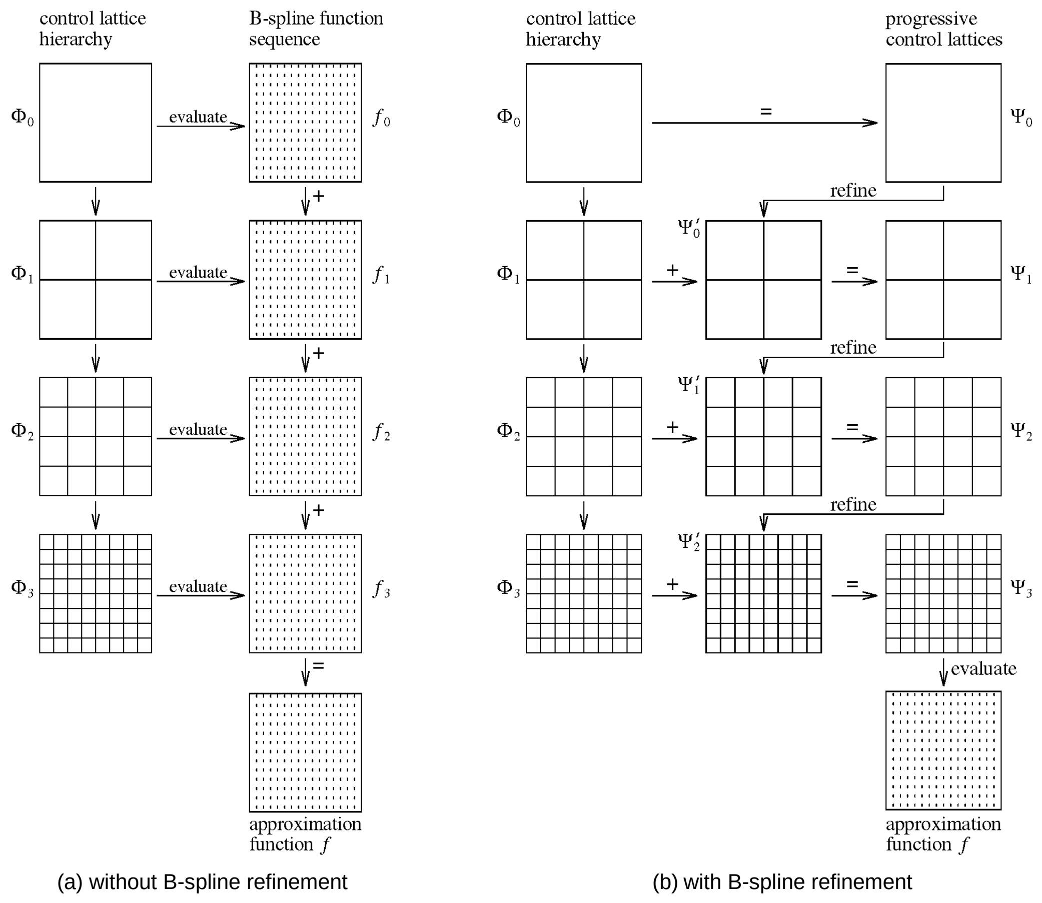

control lattice hierarchy
B-spline function sequence
Φ_0
Φ_1
Φ_2
Φ_3
evaluate
f_0
f_1
f_2
f_3
+
=
approximation function f
(a) without B-spline refinement
control lattice hierarchy
progressive control lattices
Ψ_0
Ψ_1
Ψ_2
Ψ_3
Ψ'_0
Ψ'_1
Ψ'_2
refine
evaluate
approximation function f
(b) with B-spline refinement

Fig. 5. Approximation function evaluation in the MBA algorithm.

sively derive the control lattice Ψ_h for the final approximation function $f = g_h$. This process is depicted in Fig. 5b.

There are many methods for refining a control lattice into another so that they both generate the same B-spline functions [8], [37], [3], [38], [36]. In this paper, an $(m + 3) \times (n + 3)$ control lattice Φ is always refined to a $(2m + 3) \times (2n + 3)$ lattice Φ' whose control point spacing is half as large as that of Φ. With this restriction, we can simplify the refinement algorithm for B-spline curves in [38] to derive Φ' from Φ. Let ϕ_{ij} and ϕ'_{ij} be the *ij*-th control points in Φ and Φ', respectively. Then, the position of control point $\phi'_{2i,2j}$ in Φ' coincides with that of control point ϕ_{ij} in Φ. The values for the control points in Φ' are obtained from those in Φ by

$$\begin{aligned}
\phi'_{2i,2j} &= \frac{1}{64}\big[\phi_{i-1,j-1} + \phi_{i-1,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} \\
&\quad + 6(\phi_{i-1,j} + \phi_{i,j-1} + \phi_{i,j+1}, \phi_{i+1,j}) + 36\phi_{ij}\big], \\
\phi'_{2i,2j+1} &= \frac{1}{16}\big[\phi_{i-1,j} + \phi_{i-1,j+1} + \phi_{i+1,j} + \phi_{i+1,j+1} \\
&\quad + 6(\phi_{i,j} + \phi_{i,j+1})\big], \\
\phi'_{2i+1,2j} &= \frac{1}{16}\big[\phi_{i,j-1} + \phi_{i,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} \\
&\quad + 6(\phi_{i,j} + \phi_{i+1,j})\big], \\
\phi'_{2i+1,2j+1} &= \frac{1}{4}\big[\phi_{ij} + \phi_{i,j+1} + \phi_{i+1,j} + \phi_{i+1,j+1}\big].
\end{aligned}$$

To generate control lattice Ψ_h from data P, we do not have to keep the elements of Φ_k, Ψ_k, Ψ'_k, and P_k for all k. If we apply the B-spline approximation and refinement together at each level, Ψ_h can be derived by traversing the control lattice hierarchy from the coarsest to the finest levels. In the traversal, only one variable for each of the data and control lattice sequences is sufficient to manage the computation and intermediate result. This technique, which we call the MBA algorithm, is outlined in the following pseudocode. In the algorithm, $P - F(\Phi)$ denotes the updated data $\{(x_c, y_c, \Delta^{k+1}z_c)\}$, where $P = \{(x_c, y_c, \Delta^k z_c)\}$ and $f_k = F(\Phi)$.

MBA Algorithm

```
Input: scattered data $P = \{(x_c, y_c, z_c)\}$
Output: control lattice $\Psi$
let $\Phi$ be the coarsest control lattice
let $\Psi' = \mathbf{0}$
while $\Phi$ does not exceed the finest control lattice do
    compute $\Phi$ from $P$ by the BA algorithm
    compute $P = P - F(\Phi)$
    compute $\Psi = \Psi' + \Phi$
    let $\Phi$ be the next finer control lattice
    refine $\Psi$ into $\Psi'$ whereby $F(\Psi') = F(\Psi)$ and $|\Psi'| = |\Phi|$
end
```

Let p be the number of data points in P and let $(m + 3) \times (n + 3)$ be the size of the finest control lattice Φ_h. The number of operations required for B-spline refinement is linear in the number of control points. Hence, the time complexity of the MBA algorithm is $O(hp + \frac{4}{3}mn)$. Since we use only one variable for each Φ_k, Ψ_k, Ψ'_k, and P_k, the space complexity of the algorithm is $O(p + mn)$. If we note that the depth of the hierarchy is much less than the number of control points, then the MBA algorithm runs with virtually the same time and space complexity as the BA algorithm for the finest lattice. However, the MBA algorithm generates a much smoother function shape with the same approximation error.

Suppose that the approximation function f must be known at $M \times N$ grid points on domain Ω. The processes for evaluating f in the basic and the optimal MBA algorithms are illustrated in Fig. 5. The basic MBA algorithm requires $O(hMN)$ time to evaluate f on Ω since it takes $O(MN)$ time to evaluate each function f_k across all h levels of the hierarchy. The optimal MBA algorithm, however, requires $O(\frac{4}{3}mn)$ time to apply the B-spline refinement to the control lattice hierarchy and $O(MN)$ time to evaluate the final B-spline function on Ω. Although both algorithms generate the same function f, the optimal MBA algorithm with B-spline refinement realizes large computational savings, especially when $MN \gg mn$. When f is evaluated on a 1,024 × 1,024 grid, it takes 0.92 second on a Sun SPARC10 for the optimal MBA algorithm to generate the function shown in Fig. 4b.

5 Multilevel B-Spline Interpolation

The MBA algorithm generates an approximation function f that passes near the data points P, but not necessarily through them. We now consider how the MBA algorithm can be used to interpolate the data. Recall that function f_k, for level $k > 0$ in the hierarchy, is derived to approximate and remove the residual error $\Delta^k z_c$ at each data point. The final function f is made to interpolate data P once this error goes to zero at some level k. In this section, we present a sufficient condition for control lattice Φ_k to generate a function f_k that removes any residual error. We also present an adaptive representation for the control lattice hierarchy to minimize memory overhead when the finest control lattice is required to be very dense.

5.1 Sufficient Condition for Interpolation

Let $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ be two points in data P_k. Without loss of generality, we assume that Φ_k has the same horizontal and vertical spacing between control points. We define the distance between p_1 and p_2 as $max\left(\left|\left\lfloor \frac{x_2}{s} \right\rfloor - \left\lfloor \frac{x_1}{s} \right\rfloor\right|, \left|\left\lfloor \frac{y_2}{s} \right\rfloor - \left\lfloor \frac{y_1}{s} \right\rfloor\right|\right)$, where s is the control point spacing. This distance represents the maximum number of horizontal or vertical lattice lines in Φ_k that lie between p_1 and p_2 after they have been projected onto domain Ω. We use it to define the interpolation property in terms of the control lattice density and the data distribution.

Let d be the minimum distance among all pairs of data points in P_k. If $d \geq 4$, then no two data points share a control point in their 4×4 neighborhoods. In that case, each control point in Φ_k contains at most one data point in its proximity data set. When the BA algorithm is applied to P_k with Φ_k, (5) reduces to (3) for all control points. This results in an interpolation function f_k that satisfies (2) at each data point in P_k. If $d < 3$, however, there is a control point ϕ in Φ_k that has at least two data points in its proximity data set. In that case, the effects of those points are blended together to determine the value of ϕ so that function f_k only approximates them.

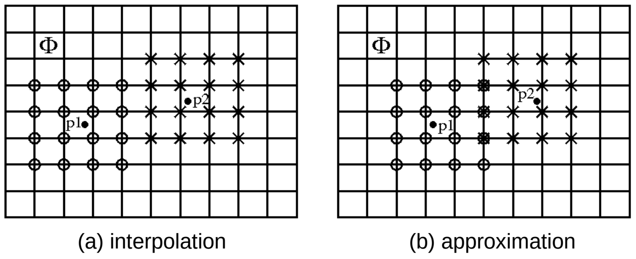


Fig. 6. Examples of the interpolation and approximation cases.

Figs. 6a and 6b show examples of the interpolation and approximation cases, respectively. Note that the interpolation property depends on the number of control lattice lines between data points, not the actual distance between them. This is depicted in Fig. 6 where points p_1 and p_2 have the same distance but different interpolation properties.

5.2 Adaptive Control Lattice Hierarchy

The interpolation condition given in Section 5.1 demonstrates that the MBA algorithm can interpolate data when the control point spacing in the finest lattice Φ_h becomes sufficiently small to satisfy $d \geq 4$. It is possible, however, that a single pair of close data points may require Φ_h to become very dense even though all other data points are sparsely distributed. This can impose large memory overhead in computing the interpolation function. To avoid this drawback, we propose an adaptive representation for the control lattice hierarchy that only stores those control points that lie in a 4×4 neighborhood about each data point.

When the B-spline approximation is applied in the MBA algorithm, control points with empty proximity data sets are assigned the value zero and do not contribute to the final function f. Therefore, to derive f, it is sufficient to maintain only those control points that have data points in their proximity data sets. We can then represent a control lattice as a set of necessary control points rather than a lattice of all control points. As successive B-spline approximations proceed across a control lattice hierarchy, many of the control points in the finer lattices have empty proximity data sets. Hence, the set representation of a control lattice makes it possible to save a lot of storage, especially for finer lattices.

At a level k in the hierarchy, the values of the necessary control points in lattice Φ_k can efficiently be computed by modifying the BA algorithm. The variables δ_{ij} and ω_{ij} are allocated in linear arrays instead of two dimensional arrays. When data points are considered in sequence, their influences on the neighboring control points are stored in the linear arrays, together with the control point indices. The arrays are then sorted by index and merged together to compute (5) for control points with nonempty proximity data sets. The control points not included in the arrays are assumed to be zero because they have empty proximity data sets. The following pseudocode reflects this modification, which we refer to as the adaptive BA algorithm.

Adaptive BA Algorithm

```
Input: scattered data $P = \{(x_c, y_c, z_c)\}$
Output: a set of control points $\{\phi_{ij}\}$
let $r = 0$
**for** each point $(x_c, y_c, z_c)$ in $P$ **do**
    set $i, j, s, t, w_{kl}, \sum\sum w_{ab}^2$ as in the BA algorithm
    **for** $k, l = 0, 1, 2, 3$ **do**
        compute $\phi_{kl}$ with (3)
        store $w_{kl}^2\phi_{kl}$ and index $(i+k, j+l)$ to $\delta_r$
        store $w_{kl}^2$ and index $(i+k, j+l)$ to $\omega_r$
        ler $r = r + 1$
    **end**
**end**
sort $\delta_r$ and $\omega_r$ by index
**for** each index $(i, j)$ in $\omega_r$ **do**
    set $r_1, r_2$ so that $index(\omega_r) = (i, j)$ for $r_1 \le r \le r_2$
    compute $\phi_{ij} = \sum_{r=r_1}^{r_2} \delta_r \Big/ \sum_{r=r_1}^{r_2} \omega_r$
**end**
```

Let p be the number of data points in P. The time complexity of the modified BA algorithm is $O(p \log p)$ because arrays δ_r and ω_r require sorting. The space complexity is $O(p)$. The time and space complexities of the algorithm depend on the number of data points, not the size of the control lattice. Hence, the algorithm will be useful at a finer level of the hierarchy where the size of the control lattice is much larger than the number of data points. Note that the set of stored control points need not form a sublattice because a lattice structure is not required for the algorithm to handle the control points.

With the set representation of a control lattice, the final function f of the MBA algorithm must be represented by the sum of the B-spline functions derived at each level of the hierarchy. Function f cannot be converted into one B-spline function on the finest control lattice because we do not maintain all of the control points in that lattice. In that case, we cannot benefit from the optimization with B-spline refinement in evaluating f although we can save a lot of storage in deriving f. A reasonable solution lies in maintaining the full control lattice at the coarser levels so that B-spline refinement can be applied, and then switching to the adaptive approach for finer control lattices. This realizes performance gains with minimal memory overhead.

6 Applications

Multilevel B-spline interpolation is well suited to any application that can be cast as a surface fitting problem. In this section, we demonstrate its use in three diverse applications: image reconstruction, image warping, and object reconstruction. The scattered data values in these cases consist of image intensities (1D), positional constraints (2D), and surface points (3D), respectively. These examples demonstrate the use of the MBA algorithm on multidimensional data for various applications.

6.1 Image Reconstruction from Nonuniform Samples

Consider a set of nonuniform samples taken from an image. Reconstruction refers to the interpolation necessary to recover the image from its samples. A robust solution to this problem is of great practical importance in image compression. Significant data reduction is possible by representing a full image with only a few scattered samples. We now address this problem by interpreting a grayscale image as a surface, where the value of each pixel represents its height. Image reconstruction from nonuniform samples is then cast into a surface fitting framework that can be solved with the MBA algorithm.

Consider the image in Fig. 7a. To derive a set of nonuniform samples, we first apply the Sobel operator [24] to the image and threshold the result to identify edges. These edge contours consist of an important set of pixels which capture the visual details of the image. Due to their rather arbitrary definition and distribution, edges alone are not sufficient to recover the image. Therefore, we also sample the image on a coarse regular grid to properly reconstruct the parts of the image where there are no nearby edge points. For the purpose of this example, we used a uniform sampling rate of one sample per six pixels along each direction. The positions of the sampled pixels are shown in Fig. 7b. The MBA algorithm is applied to the samples to reconstruct the surface in Fig. 7c. The reconstructed image that corresponds to this surface is shown in Fig. 7d.

Large data reduction is achieved here. The dimensions of the original image in Fig. 7a is 360×243. The number of sample points in Fig. 7b is 7,301, of which 4,984 lie upon the edge contours. Therefore, sample points constitute only 8.3 percent of the total number of pixels in the original image. In the MBA algorithm, we used $m_0 = 2$, $n_0 = 1$ and $m_h = 360$, $n_h = 243$ for the size of the coarsest and finest control lattices, respectively. Fig. 7d demonstrates that the MBA algorithm can adequately reconstruct an image from a proper set of samples. The original and reconstructed images are visually indistinguishable.

6.2 Image Warping

Image warping deals with the geometric transformation of digital images. It requires a warp function to establish a spatial transformation between the input and output images. Depending on the application, warp functions may take on many different forms. Simple transformations may be specified by analytic expressions including affine, projective, bilinear, and polynomial transformations. More sophisticated warp functions that are not conveniently expressed in analytic terms can be determined from a sparse collection of feature points which are displaced to define a transformation. The displacements for the remaining points are evaluated through interpolation.

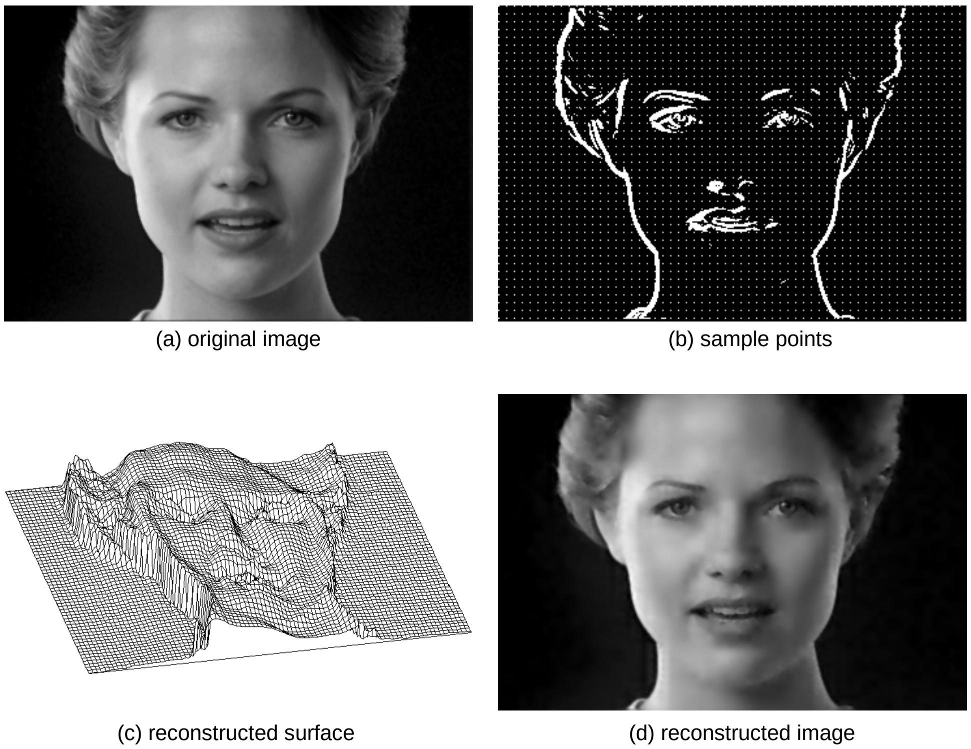


Fig. 7. The MBA algorithm is applied to nonuniform image samples to generate a visually indistinguishable reconstructed image.

A warp function relates the (x, y) points in the input (original) image to their (x', y') counterparts in the output (warped) image. We can define the mapping in terms of two functions: $x' = X(x, y)$ and $y' = Y(x, y)$. That is, for every point (x, y) in the input image, functions X and Y relate its corresponding x'- and y'-coordinates, respectively. Consider feature points (x_c, y_c) in the input image and their displaced positions (x'_c, y'_c) in the output image. The MBA algorithm can be used to determine a warp function from the feature point displacements by constructing two smooth surfaces for functions X and Y. One surface for X is required to pass through points (x_c, y_c, x'_c), while the other surface for Y passes through (x_c, y_c, y'_c).

An image warping example is shown Fig. 8. The original image is given in Fig. 8a. Fig. 8b shows the user-specified features consisting of points placed on characteristic parts of the face, such as the profile, eyes, nose, and mouth. These feature points are moved to new positions to warp the original image. Fig. 8c shows the new positions and the deformation due to the warp function derived by the MBA algorithm. The warped image produced by resampling the original image using the derived warp function is given in Fig. 8d.

The dimensions of the original image in Fig. 8a is 360×243. The number of feature points in Fig. 8b is 242. Two functions for the x'- and y'-components of the warp function are derived simultaneously by using the 2D values of control points. In the MBA algorithm, we used $m_0 = 2$, $n_0 = 1$ and $m_h = 360$, $n_h = 243$ for the size of the coarsest and finest control lattices, respectively. Fig. 8 demonstrates that the MBA algorithm can adequately generate smooth warp functions from a few selected feature points.

6.3 Object Reconstruction

Object reconstruction from a set of scattered 3D points is important in geometric modeling. The MBA algorithm can be used to solve this problem when a mapping between the scattered points and a 2D parametric space is defined. In that case, each control point in the algorithm is associated with a 3D value corresponding to the x-, y-, and z-coordinates of a point on the object. The reconstructed object is a 3D uniform cubic B-spline surface that passes through the scattered points.

Fig. 9 illustrates an example. Fig. 9a shows a panoramic image of cylindrical range data of a head, as measured by a range finder (Cyberware 4020/PS 3-D Digitizer). The pixel intensities depict depth values. We convert the image to a 3D object by mapping each pixel to a 3D point, as shown in Fig. 9b. To derive a set of scattered points on the object, we first apply the Sobel operator [24] to the image in Fig. 9a and threshold the result. The 3D points corresponding to the resulting pixels consist of characteristic parts of the object that have high curvature. We also sample a coarse regular grid (e.g., one out of every ten points along each direction) on the object to handle low curvature regions. Fig. 9c shows the set of sampled points. The MBA algorithm is applied to the samples to reconstruct the object in Fig. 9d.

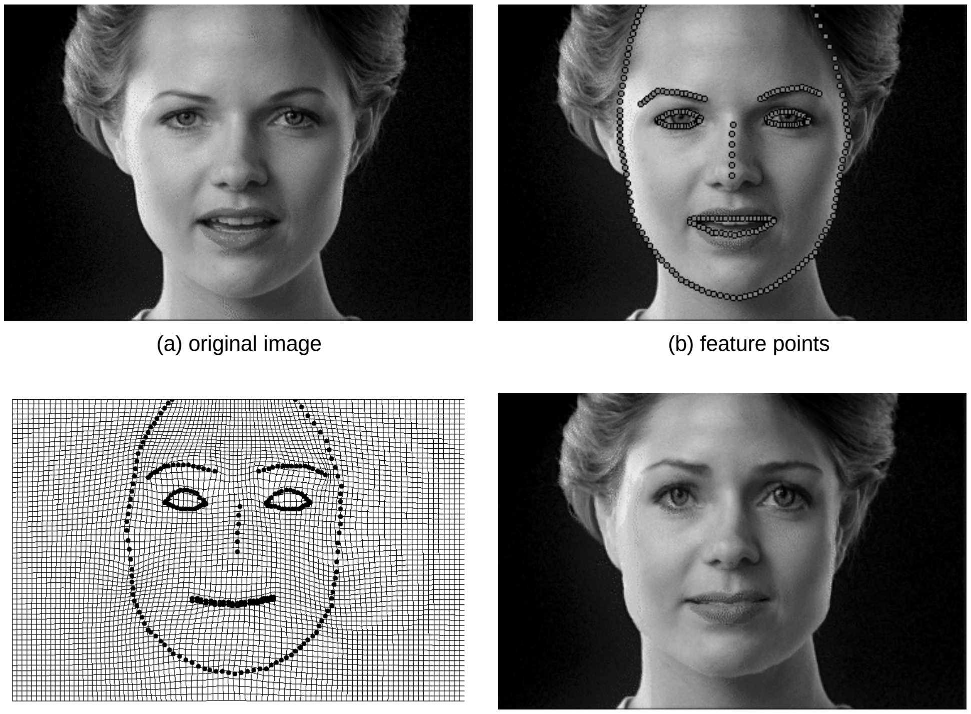


Fig. 8. The MBA algorithm is applied to scattered feature constraints to produce a smooth warp function for image warping.

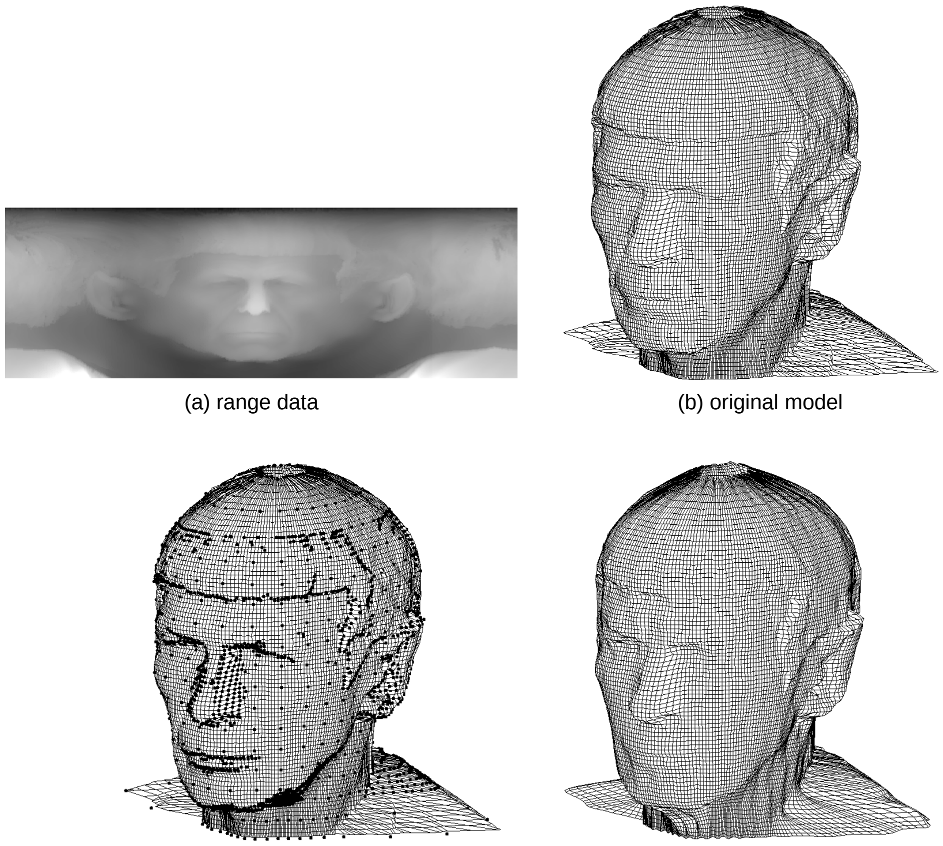


Fig. 9. The MBA algorithm is applied to nonuniform range data samples to reconstruct a 3D object.

TABLE 1
PERFORMANCE DATA (TIME MEASUREMENTS IN SECONDS)

	Basic MBA	MBA	Basic adaptive MBA	Adaptive MBA	Multigrid relaxation
Surface construction	4.18	0.92	-	-	26.62
Image reconstruction	3.86	2.23	-	-	4.15
Image warping	2.14	1.65	1.76	0.98	7.90
Object reconstruction	6.54	3.77	6.24	4.38	12.23

The 512×170 image in Fig. 9a was used as the 2D parametric space to apply the MBA algorithm to the sample points. The number of samples in Fig. 9c is 4,112. In the MBA algorithm, we used $m_0 = 2$, $n_0 = 1$ and $m_h = 512$, $n_h = 170$ for the size of the coarsest and finest control lattices, respectively. Fig. 9 demonstrates the potential of the MBA algorithm in reconstructing an object from a set of scattered 3D points.

7 DISCUSSION

In this section, we first present experimental results for the performance and reconstruction accuracy of the MBA algorithm. We provide insight into the role of B-spline refinement and adaptive control lattice hierarchy in the measured performance. The results are compared to that of multigrid relaxation for thin plate splines, an efficient method for scattered data interpolation. In all cases, the MBA algorithm is faster than multigrid relaxation. We then consider the use of an affine approximation as a preprocessing of multilevel B-spline approximation. Finally, we compare the MBA algorithm to hierarchical B-spline refinement, a related technique presented for object modeling.

7.1 Performance

The performance of the MBA algorithm for the examples in this paper are given in Table 1. Four versions of the algorithm were tested:

- basic MBA algorithm (without B-spline refinement)
- MBA algorithm (with B-spline refinement)
- basic adaptive MBA algorithm (without B-spline refinement)
- adaptive MBA algorithm (with B-spline refinement)

The first row was derived from the surface construction example shown in Fig. 4b. The other rows were obtained from the examples in Section 6. The performance numbers in Table 1 were measured in seconds on a Sun SPARC10.

In the basic adaptive MBA algorithm applied to a control lattice hierarchy, the adaptive BA algorithm is used to derive lattices Φ_k for which $16p < |\Phi_k|$, where p is the number of data points. The final approximation function f is computed by adding the functions f_k from each control lattice Φ_k as in the basic MBA algorithm. When Φ_k is available as a lattice, we use the forward difference method [14], [51] to compute f_k. If Φ_k is stored by a set of control points, however, f_k must only be computed in the neighborhoods of those control points. In that case, we use a lookup table for B-spline basis functions to efficiently compute f_k in those neighborhoods. The basic adaptive and adaptive MBA algorithms differ only in the use of B-spline refinement for reducing the coarser control lattices into one equivalent lattice. Note that B-spline refinement can be applied only to the coarser control lattices that violate the condition $16p < |\Phi_k|$. This realizes computational savings in evaluating the final approximation function.

The first row in Table 1 shows that the performance of the MBA algorithm is far superior to that of the basic MBA algorithm in constructing the surface in Fig. 4b. This demonstrates the important role of B-spline refinement in realizing performance gains. In this case, the adaptive MBA algorithms were not applied because the data points are sparse and can be interpolated with a rather coarse 64×64 control lattice. The second row in Table 1 shows that similar performance gains due to B-spline refinement were realized in the image reconstruction example. The adaptive MBA algorithms were also not applied in this case because the large number of data points violated the condition $16p < |\Phi_k|$ for every control lattice Φ_k.

In the image warping example, the adaptive MBA algorithm was the fastest among all tested. In that algorithm, the approximation functions f_k are computed efficiently on the coarser control lattices by using B-spline refinement. On the finer control lattices, the adaptive BA algorithm is used and the approximation functions are efficiently computed by applying a lookup table for B-spline basis functions to the set of necessary control points. In this case, the MBA algorithm is slower because it must apply B-spline refinement to a large number of control points on the finer lattices. In general, if the number of data points is small and the control lattice size is large, B-spline function evaluation using a lookup table can be more efficient than B-spline refinement. Hence, the adaptive MBA algorithm outperforms the MBA algorithm on the finer control lattices in spite of the overhead of sorting the arrays δ_r and ω_r to compute the necessary control point values.

The basic adaptive MBA algorithm is slightly slower than the MBA algorithm in the image warping example. Although the basic adaptive MBA algorithm efficiently computes the approximation functions at the finer levels, the performance loss is due to the computation required to densely evaluate the functions at the coarser levels rather than applying B-spline refinement. The difference between the numbers for the two adaptive MBA algorithms shows that B-spline refinement is very helpful for the coarser levels.

The last row in Table 1 shows that the adaptive MBA algorithm is slower than the MBA algorithm in the object reconstruction example. In this case, the arrays δ_r and ω_r are very large because there are a large number of data points. Sorting these large arrays requires a computation overhead that undermines the efficiency of the adaptive MBA algorithm at the finer levels. Actually, the evaluation of an approximation function using a lookup table at a finer

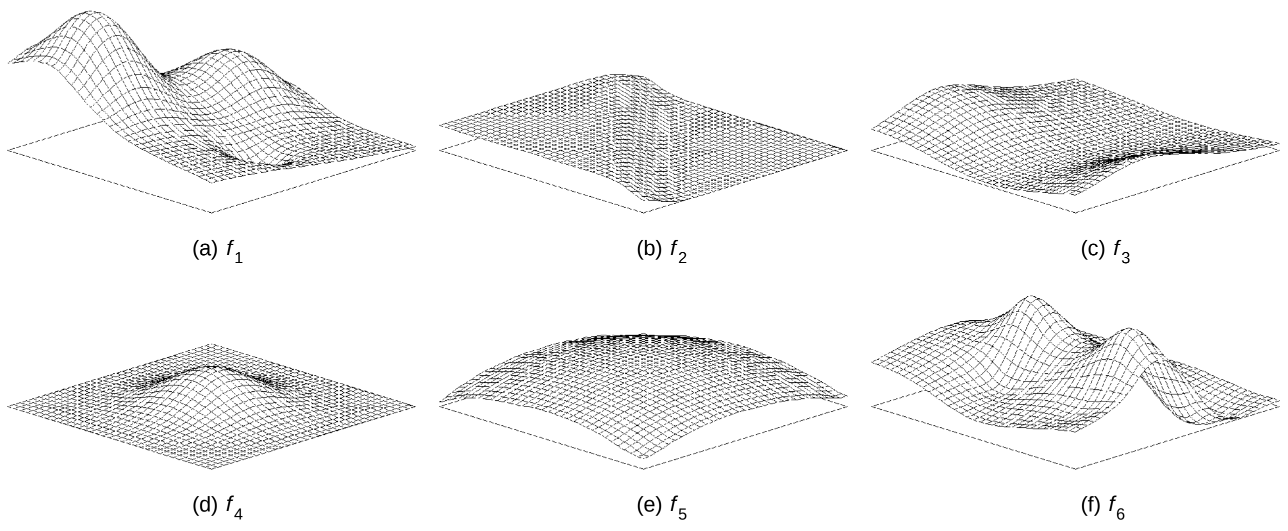


Fig. 10. Test functions used for demonstrating reconstruction accuracy.

level is not very efficient in this case due to the large number of data points.

In the last column of Table 1, the performance data is derived by using multigrid relaxation to obtain thin plate splines for the examples in this paper. The multigrid relaxation method has been considered an efficient numerical technique to solve a linear system [5]. The method was used in early vision problems to accelerate the numerical solution of thin plate surfaces that interpolate scattered data points [48], [49]. An alternate approach using hierarchical basis functions or wavelets has been proposed to provide an efficient solution to scattered data interpolation with thin plate splines [46], [25]. In the approach, a set of hierarchical basis functions is used to accelerate the convergence of an iterative numerical technique such as conjugate gradient descent.

Let an interpolation function be evaluated on a grid of size $M \times N$. The time complexity of the multigrid relaxation method is $O(MN)$, while the approach using hierarchical basis functions runs in $O(MN \log MN)$. When the grid is large, we can expect that multigrid relaxation derives a thin plate surface faster than hierarchical basis functions. Hence, multigrid relaxation for thin plate splines was chosen to be compared to the MBA algorithm in Table 1. To implement the multigrid relaxation method, we followed the pseudocode in [5], where the parameters v_0, v_1, and v_2 determine the number of relaxations. A summary of this approach to derive thin plate splines can be found in [32]. We used $v_0 = v_1 = v_2 = 1$ for the surface construction example and $v_0 = 1$ and $v_1 = v_2 = 3$ for the other examples.

The data in Table 1 shows that multigrid relaxation for thin plate splines is slower than any version of the MBA algorithm. The performance of multigrid relaxation strongly depends on the size of the underlying grid and remains nearly constant regardless of the number of data points. In the surface construction example, the approximation function is evaluated on a 1,024 × 1,024 grid and multigrid relaxation is several times slower than the MBA algorithm. In the image reconstruction example, the performance gap is smaller than in the other examples due to the large number of data points.

7.2 Reconstruction Accuracy

To demonstrate the accuracy of reconstruction by the MBA algorithm, we performed experiments with several test functions. Given a test function $f(x, y)$, we first sampled data points from it and applied the MBA algorithm to obtain an approximation function g. The difference between g and f is then measured by computing the root mean square error between the function values on a dense grid.

We selected five out of the six test functions used in Nielson's experiment [41] for trivariate scattered data interpolation. We simplified the selected functions to make them bivariate functions. The resulting test functions are

$$f_1(x,y) = 0.75\exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] + 0.75\exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right]$$
$$+\,0.5\exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] - 0.2\exp\left[-(9x-4)^2-(9y-7)^2\right],$$

$$f_2(x,y) = \left(\tanh(9-9x-9y)+1\right)\big/9,$$

$$f_3(x,y) = \left(1.25+\cos(5.4y)\right)\big/\left(6+6(3x-1)^2\right),$$

$$f_4(x,y) = \exp\left[-\frac{81}{4}\left((x-0.5)^2+(y-0.5)^2\right)\right]\Big/3,$$

$$f_5(x,y) = \sqrt{64-81\left((x-0.5)^2+(y-0.5)^2\right)}\Big/9 - 0.5\ .$$

In addition, we used a bicubic B-spline surface f_6 as a test function. Fig. 10 shows the test functions. The domain of the functions is $\{(x,y) \mid 0 \le x \le 1,\ 0 \le y \le 1\}$. The approximate ranges of the functions are f_1:$[0.0, 1.3]$, f_2:$[0.0, 0.22]$, f_3:$[0.0, 0.38]$, f_4:$[0.0, 0.34]$, f_5:$[0.0, 0.39]$, f_6:$[-0.1, 1.0]$.

For each test function, we used four data sets, M100, M500, L160, and C160. M100 and M500 are small and large data sets, which consist of 100 and 500 points, respectively. We uniformly sampled 7 × 7 and 15 × 15 data points for M100 and M500, respectively, while the other points were randomly sampled. L160 consists of data points sampled from several lines, which is similar to sampling from the edges of an object. We used eight lines, and 20 points were sampled and perturbed on each of the lines. In C160, the sampled points were clustered, which is analogous to real-world sampling [41]. C160 consists of eight clusters, each of which has 20 points. Fig. 11 shows the sampling positions for the data sets.

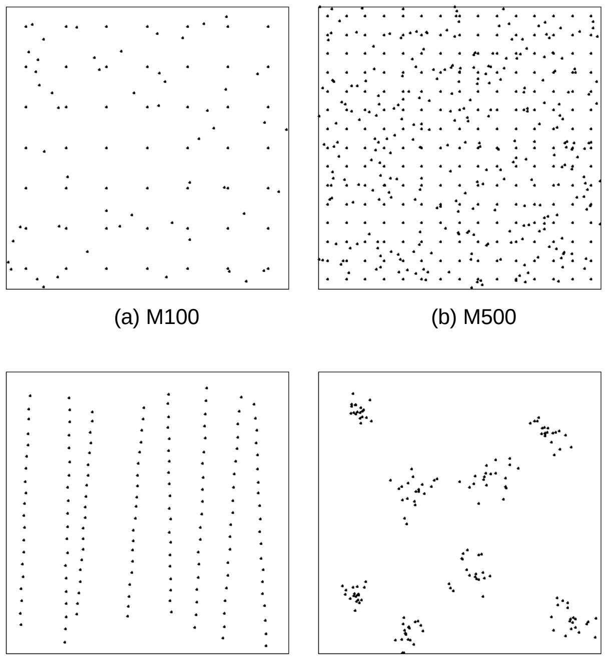


Fig. 11. Sampling positions for test data sets.

For each test function *f*, we derived an approximation function *g* by applying the MBA algorithm to each of the data sets, M100, M500, L160, and C160. The root mean square (RMS) error between *f* and *g* was measured on a dense grid over the domain. That is,

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

where $x_i = i/M, y_j = j/N$, and $M = N = 50$. To normalize the error values, we divided the RMS error by the difference of the maximum and minimum values of *f* over the domain. Table 2 shows the results.

TABLE 2
NORMALIZED RMS ERRORS BETWEEN TEST FUNCTIONS AND THEIR APPROXIMATIONS

	f_1	f_2	f_3	f_4	f_5	f_6
M100	0.016	0.025	0.013	0.006	0.027	0.016
M500	0.001	0.005	0.003	0.0008	0.007	0.002
L160	0.031	0.032	0.042	0.008	0.049	0.022
C160	0.082	0.097	0.130	0.086	0.080	0.081

Table 2 demonstrates that the MBA algorithm reconstructs a test function quite accurately regardless of the type of the function. The normalized RMS error is less than 5 percent of the range of the function values, except for the case of cluster sampling. It is clear from the examples of M100 and M500 that the accuracy increases when the number of data points gets larger.

7.3 Initial Linear Approximation

Only at the coarsest level $k = 0$ does data P_0 consist of the original data points *P* in the MBA algorithm. At each successively finer level $k > 0$, data P_k contains the residual error between the intermediate approximation function g_{k-1} and the original data *P*. The BA algorithm applied at level *k* attempts to remove the residual error by deriving a B-spline approximation of P_k that may be added to g_{k-1} to yield a more accurate approximation g_k. In that case, if the proximity data set of a control point ϕ_{ij} is empty, ϕ_{ij} has no effect in reducing the residual errors. Then, the most natural choice for the value of ϕ_{ij} is zero so that the approximation from the coarser levels remains unaffected. When ϕ_{ij} has a nonempty proximity data set, ϕ_{ij} is determined by a weighted average of the values ϕ_c computed by (4). Among the solutions of (2), we chose (4) to minimize the deviation of a B-spline approximation from zero. This serves to retain the approximation from the coarser levels as much as possible while properly reducing the residuals.

Regardless of the proximity data set of a control point, the BA algorithm determines its value so as to minimize the changes to the approximation function obtained from the coarser levels. This is intuitive for all but the coarsest level, which must provide a good initial approximation of the data. In general, the BA algorithm applied at the coarsest level generates a reasonable global approximation of data points. In the event that the coarsest control lattice is very fine, many of its control points may have empty proximity data sets and be assigned zero. The final approximation will thereby tend to contain local peaks near the data points, which may be unsatisfactory for many applications (see Fig. 4c). In this case, we may improve matters by using a least squares linear fit to obtain a better initial approximation.

We now review how to compute a linear approximation function f_{-1} from given data points, $P = \{(x_c, y_c, z_c) \mid c = 1, \ldots, p\}$. Let

$$f_{-1}(x, y) = a_1 x + a_2 y + a_3, \tag{6}$$

where a_1, a_2, and a_3 are parameters. We may fit plane f_{-1} to data *P* by solving for **x** in the set of linear equations

$$\mathbf{Ax} = \mathbf{b},$$

where

$$\mathbf{A} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_p & y_p & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}, \quad \text{and } \mathbf{b} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_p \end{pmatrix}.$$

We derive the least-squared solution for f_{-1} that minimizes the error $\sum_c (z_c - f_{-1}(x_c, y_c))^2$. The solution can be computed by simple linear algebra [42] using the pseudoinverse of **A**:

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}, \tag{7}$$

where $\mathbf{A}^T$ is the transpose of matrix **A**. The product $\mathbf{A}^T\mathbf{A}$ yields a 3×3 matrix whose inverse can be computed efficiently. The resulting function f_{-1} is the best linear approximation to data *P* in terms of the squared approximation errors.

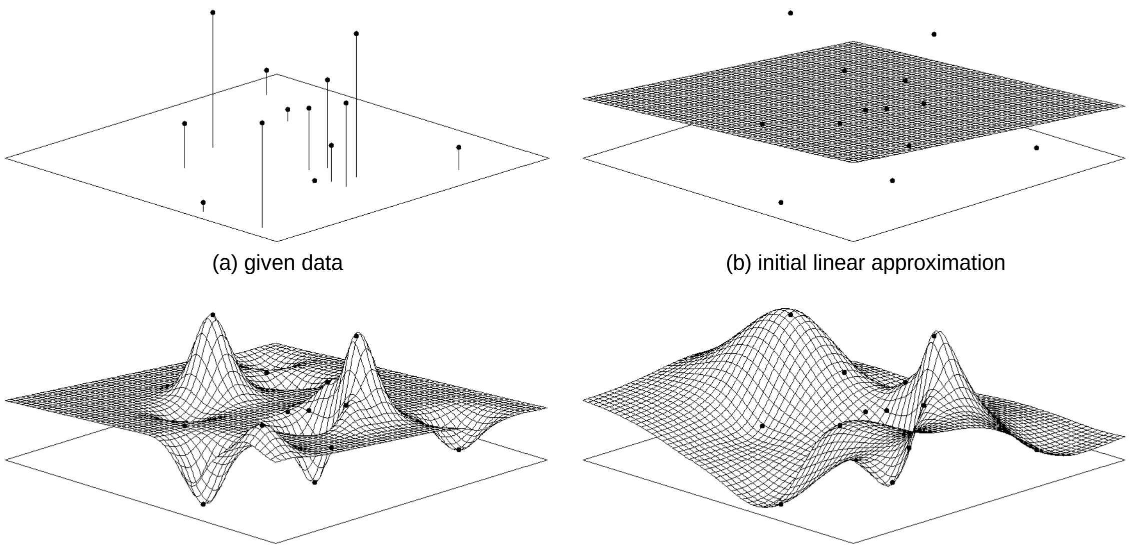

(a) given data
(b) initial linear approximation
(c) $m_0 = n_0 = 16, m_h = n_h = 64$
(d) $m_0 = n_0 = 1, m_h = n_h = 64$

Fig. 12. Reconstructed surfaces using the MBA algorithm with an initial linear approximation.

When we use an initial linear approximation, we first derive function f_{-1} from the original data P by (7). Then, the MBA algorithm is applied to the deviations P_0 of f_{-1} from P, where $P_0 = \{(x_c, y_c, \Delta^0 z_c) \mid \Delta^0 z_c = z_c - f_{-1}(x, y)\}$. The final approximation function is the sum of f_{-1} and the function obtained by the MBA algorithm from P_0. With this procedure, every level in the MBA algorithm, including the coarsest level, approximates the deviations between the current intermediate function and the data points. This justifies the use of a minimal norm solution (4) and assigning zero to control points having empty proximity data sets in the BA algorithm.

Fig. 12 demonstrates the role of a least squares linear fit as an initial approximation in the MBA algorithm. Fig. 12a shows the same data given in Fig. 4a. Fig. 12b depicts the linear approximation function f_{-1} fitted to the data. In Fig. 12c, the initialization is applied to the case in Fig. 4c. The approximation function in Fig. 12c continues to have peaks and valleys near the data points, as in Fig. 4c. Note, however, that the function values are no longer zero away from the data. They are now derived from the underlying plane f_{-1}. Fig. 12d shows the result of applying the linear initialization to the case in Fig. 4b. The resulting approximation function remains virtually the same as that in Fig. 4b. In this case, the coarsest level in the MBA algorithm generates a reasonable global approximation of data points, and the linear fit is not warranted.

7.4 Comparison to Hierarchical B-Spline Refinement

Forsey and Bartels proposed hierarchical B-spline refinement in [16] to gain finer control in editing a surface. The purpose of the control lattice hierarchy is to restrict refinement to the locality at which an editing effect is desired. That paper, however, did not describe how to manipulate the hierarchy to interpolate a set of data points. To address this problem, they presented a technique in [17], [18] that augments B-spline approximation with hierarchical B-spline refinement to interpolate a *grid* of data using a control lattice hierarchy. Their method, however, cannot handle *scattered* data.

In this paper, we presented multilevel B-splines to handle scattered data. Instead of locally refining control lattices, we maintain all control points in the hierarchy to apply B-spline refinement. This allows us to reduce a sequence of B-spline functions, defined by a control lattice hierarchy, as *one* equivalent B-spline function, defined by *one* control lattice. In addition to the performance gains in evaluating the approximation function, this reduction makes the final approximation function suitable for any conventional modeling and rendering system. In contrast, the hierarchical B-spline refinement technique needs a complicated data structure to maintain the local refinements of control lattices. In this paper, even when an adaptive control lattice hierarchy is used, a control lattice can be represented by a simple set of necessary control points.

8 Conclusion and Future Work

This paper has presented a fast approximation and interpolation algorithm for scattered multivariate data. The algorithm is based on B-spline approximation. A B-spline control lattice is efficiently determined by minimizing a local approximation error for each control point. The resulting C^2-continuous function passes near densely distributed data points and interpolates isolated points. However, a tradeoff exists between the shape smoothness and approximation accuracy of the function, depending on the control lattice density.

Multilevel B-spline approximation was introduced to circumvent this tradeoff. The algorithm makes use of a hierarchy of control lattices to generate a sequence of functions whose sum approaches the desired approximation function. In the sequence, a function from the coarsest control lattice provides an initial estimate, which is further refined in accuracy by functions derived at finer levels. Large performance gains are realized by applying B-spline refinement to reduce the sum of these functions into one equivalent B-spline function.

We also presented a sufficient condition for multilevel B-spline approximation to generate an interpolation function through scattered data points. Interpolation is achieved when the control point spacing in the finest lattice becomes sufficiently small relative to the data distribution. We proposed an adaptive representation of the control lattice hierarchy to minimize the memory overhead that may be introduced at finer levels.

In his work comparing several scattered data interpolation methods, Franke used accuracy, visual aspects, sensitivity to parameters, timing, storage requirements, and ease of implementation as the tested characteristics of the methods [20]. In Section 7.1 and Section 7.2, we showed that the MBA algorithm has good performance with respect to the timing and accuracy characteristics, respectively. In terms of visual aspects, the algorithm generates an approximation function with a smooth shape. The parameters of the algorithm are the sizes of the coarsest and finest control lattices in the control lattice hierarchy. The effects of those parameters are quite intuitive as illustrated in Fig. 4. The storage requirement of the MBA algorithm relates to the size of the finest control lattice, which can be controlled by the user. As demonstrated by the pseudocodes in this paper, the MBA algorithm is very easy to implement.

The multilevel B-spline approximation can reconstruct a planar surface when we apply an initial linear approximation to the data, as described in Section 7.3. When data points are sampled from a nonplanar surface (e.g., a piecewise bicubic surface), the algorithm does not exactly reconstruct the original surface. However, the algorithm is guaranteed to generate a C^2-continuous piecewise bicubic surface interpolating the data points, which is expected to be a nice approximation of the original surface. On the other hand, the algorithm is not affine-invariant. Since data points are projected to underlying lattices of varying resolution, the relative positioning between the data points and the lattices affects the approximation function. In future work, we will conduct a complete analysis of the reconstruction class of the algorithm and investigate modifications to satisfy the affine-invariance property.

The scattered data interpolation technique introduced here can be applied to other areas of computer graphics. We have demonstrated its use for warp generation in image warping. This has direct impact on image morphing as well. Furthermore, the algorithm may play a significant role in data compression. We have demonstrated that high fidelity image and object reconstruction is possible from a selected set of sparse and irregular samples. Future work will address techniques to uniquely determine a minimal set of samples necessary to achieve high quality reconstruction within a specified error tolerance.

ACKNOWLEDGMENTS

Dr. Seungyong Lee performed work on this paper while he was a visiting scholar in the Department of Computer Science at the City College of New York. This work was supported in part by U.S. National Science Foundation Presidential Young Investigator award IRI-9157260, PSC-CUNY grant RF-666338, and the Korean Ministry of Science and Technology (contract 94-S-05-A-03 of STEP 2000).

REFERENCES

[1] E. Arge, M. Dæhlen, and A. Tveito, “Approximation of Scattered Data Using Smooth Grid Functions,” *J. Computational and Applied Math.*, vol. 59, pp. 191-205, 1995.

[2] R. Barnhill, “Representation and Approximation of Surfaces,” J.R. Rice, ed., *Mathematical Software III*, pp. 68-119. New York: Academic Press, 1977.

[3] W. Bohm and H. Prautzsch, “The Insertion Algorithm,” *Computer Aided Design*, vol. 17, no. 2, pp. 58-59, 1985.

[4] A. Brandt, “Multi-Level Adaptive Solutions to Boundary-Value Problems,” *Mathematics of Computation*, vol. 31, no. 138, pp. 333-390, 1977.

[5] W.L. Briggs, *A Multigrid Tutorial*. Lancaster, Penn.: SIAM, Lancaster Press, 1987.

[6] P.J. Burt, “Moment Images, Polynomial Fit Filters, and the Problem of Surface Interpolation,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 144-152, 1988.

[7] R. Clough and J. Tocher, “Finite Element Stiffness Matrices for Analysis of Plates in Bending,” *Proc. Conf. Matrix Methods in Structural Mechanics*, pp. 515-545, 1965.

[8] E. Cohen, T. Lyche, and R. Riesenfeld, “Discrete B-Splines and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics,” *Computer Graphics and Image Processing*, vol. 14, no. 2, pp. 87-111, 1980.

[9] I. Crain and K. Bhattacharyya, “Treatment of Nonequispaced Two-Dimensional Data with a Digital Computer,” *Geoexploration*, vol. 5, pp. 173-194, 1967.

[10] G. Cressman, “An Operational Objective Analysis System,” *Mon. Wea. Rev.*, vol. 87, pp. 367-374, 1959.

[11] J. Duchon, “Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces,” C. Chui, L. Schumaker, and J. Ward, eds., *Multivariate Approximation Theory*, pp. 85-100. Basel, Switzerland: Birkhauser, 1975.

[12] N. Dyn, “Interpolation and Approximation by Radial and Related Function,” C. Chui, L. Schumaker, and J. Ward, eds., *Approximation Theory VI*, pp. 211-234. San Diego, Calif.: Academic Press, 1989.

[13] N. Dyn, D. Levin, and S. Ruppa, “Numerical Procedures for Global Surface Fitting of Scattered Data by Radial Functions,” *SIAM J. Sci. Statis. Comput.*, vol. 7, pp. 639-659, 1986.

[14] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice,* 2nd ed. Reading, Mass.: Addison-Wesley, 1990.

[15] T.A. Foley and G.M. Nielson, “Multivariate Interpolation to Scattered Data Using Delta Iteration,” *Approximation Theory III*, E. Cheney, ed., pp. 419-424. New York: Academic Press, 1980.

[16] D.R. Forsey and R.H. Bartels, “Hierarchical B-Spline Refinement,” *Computer Graphics (Proc. SIGGRAPH ‘88)*, vol. 22, no. 4, pp. 205-212, 1988.

[17] D.R. Forsey and R.H. Bartels, “Tensor Products and Hierarchical Fitting,” *Curves and Surfaces in Computer Vision and Graphics II (Proc. SPIE 1610)*, pp. 88-96, 1991.

[18] D.R. Forsey and R.H. Bartels, “Surface Fitting with Hierarchical Splines,” *ACM Trans. Graphics*, vol. 14, no. 2, pp. 134-161, 1995.

[19] R. Franke, “A Critical Comparison of Some Methods for Interpolation of Scattered Data,” Technical Report NPS-53-79-003, Naval Postgraduate School, 1979.

[20] R. Franke, “Scattered Data Interpolation: Tests of Some Methods,” *Math. Comp.*, vol. 38, pp. 181-200, 1982.

[21] R. Franke and G.M. Nielson, “Scattered Data Interpolation of Large Sets of Scattered Data,” *Intl. J. Numerical Methods in Eng.*, vol. 15, pp. 1,691-1,704, 1980.

[22] R. Franke and G.M. Nielson, “Scattered Data Interpolation and Applications: A Tutorial and Survey,” *Geometric Modelling: Methods and Their Application*, H. Hagen and D. Roller, eds., pp. 131-160. Berlin: Springer-Verlag, 1991.

[23] A. Glassner, *Principles of Digital Image Synthesis.* San Francisco, Calif.: Morgan Kaufmann, 1995.

[24] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*. Reading, Mass.: Addison-Wesley, 1992.

[25] S.J. Gortler and M.F. Cohen, "Hierarchical and Variational Geometric Modeling with Wavelets," *Symp. Interactive 3D Graphics*, pp. 35-42, 1995.

[26] A. Goshtasby, "Piecewise Cubic Mapping Functions for Image Registration," *Pattern Recognition*, vol. 20, no. 5, pp. 525-533, 1987.

[27] R. Hardy, "Multiquadratic Equations of Topography and Other Irregular Surfaces," *J. Geophysical Research*, vol. 76, no. 8, pp. 1,905-1,915, 1971.

[28] J. Hoschek and D. Lasser, *Computer Aided Geometric Design*. Wellesley, Mass.: A.K. Peters, Ltd., 1993.

[29] W.M. Hsu, J.F. Hughes, and H. Kaufman, "Direct Manipulation of Free-Form Deformations," *Computer Graphics (Proc. SIGGRAPH '92)*, vol. 26, no. 2, pp. 177-184, 1992.

[30] C. Lawson, "Software for C^1 Surface Interpolation," *Mathematical Software III*, J.R. Rice, ed., pp. 161-194. New York: Academic Press, 1977.

[31] S. Lee, K.-Y. Chwa, J. Hahn, and S.Y. Shin, "Image Morphing Using Deformable Surfaces," *Proc. Computer Animation '94*, pp. 31-39, 1994.

[32] S. Lee, K.-Y. Chwa, J. Hahn, and S.Y. Shin, "Image Morphing Using Deformation Techniques. *J. Visualization and Computer Animation*, vol. 7, no. 1, pp. 3-23, 1996.

[33] S. Lee, K.-Y. Chwa, S.Y. Shin, and G. Wolberg, "Image Metamorphosis Using Snakes and Free-Form Deformations," *Computer Graphics (Proc. SIGGRAPH '95)*, pp. 439-448, 1995.

[34] S. Lee, G. Wolberg, K.-Y. Chwa, and S.Y. Shin, "Image Metamorphosis with Scattered Feature Constraints," *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 4, pp. 337-354, 1996.

[35] P. Litwinowicz and L. Williams, "Animating Images with Drawings," *Computer Graphics (Proc. SIGGRAPH '94)*, pp. 409-412, 1994.

[36] T. Lyche, "Note on the Oslo Algorithm," *Computer Aided Design*, vol. 20, no. 6, pp. 353-355, 1988.

[37] T. Lyche, E. Cohen, and K. Morken, "Knot Line Refinement Algorithms for Tensor Product B-Spline Surfaces," *Computer Aided Geometric Design*, vol. 2, pp. 133-139, 1985.

[38] T. Lyche and K. Morken, "Making the Oslo Algorithm More Efficient," *SIAM J. Numerical Analysis*, vol. 23, no. 3, pp. 663-675, 1986.

[39] D. McLain, "Two Dimensional Interpolation from Random Data," *Comp. J.*, vol. 19, pp. 178-181, 1976.

[40] D.P. Mitchell, "Generating Antialiased Images at Low Sampling Densities," *Computer Graphics (Proc. SIGGRAPH '87)*, vol. 21, no. 4, pp. 65-72, 1987.

[41] G.M. Nielson, "Scattered Data Modeling," *IEEE Computer Graphics and Applications*, vol. 13, no. 1, pp. 60-70, 1993.

[42] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, England: Cambridge Univ. Press, 1992.

[43] D. Ruprecht and H. Muller, "Image Warping with Scattered Data Interpolation," *IEEE Computer Graphics and Applications*, vol. 15, no. 2, pp. 37-43, Mar. 1995.

[44] L. Schumaker, "Fitting Surfaces to Scattered Data," *Approximation Theory II*, C. Chui, L. Schumaker, and G. Lorentz, eds., pp. 203-268. New York: Wiley, 1976.

[45] D. Shepard, "A Two Dimensional Interpolation Function for Irregularly Spaced Data," *Proc. ACM 23rd Nat'l Conf.*, pp. 517-524, 1968.

[46] R. Szeliski, "Fast Surface Interpolation Using Hierarchical Basis Functions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 513-528, 1990.

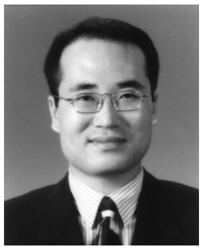
[47] D. Terzopoulos, "Multilevel Computational Processes for Visual Surface Reconstruction," *Computer Vision, Graphics, and Image Processing*, vol. 24, pp. 52-96, 1983.

[48] D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 129-139, 1986.

[49] D. Terzopoulos, "Regularization of Inverse Visual Problems Involving Discontinuities," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 413-424, 1986.

[50] W. Welch and A. Witkin, "Variational Surface Modeling," *Computer Graphics (Proc. SIGGRAPH '86)*, vol. 26, no. 2, pp. 157-166, 1992.

[51] G. Wolberg. *Digital Image Warping*. Los Alamitos, Calif.: IEEE CS Press, 1990.



Seungyong Lee received his BS degree in computer science and statistics from Seoul National University, Korea, in 1988, and his MS and PhD degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1990 and 1995, respectively. He worked at the City College of New York as a visiting scholar from March 1995 to August 1996. He is currently an assistant professor in the Department of Computer Science and Engineering at Pohang University of Science and Technology (POSTECH), Korea. His research interests include computer graphics, computer animation, and image processing.

George Wolberg received his BS and MS degrees in electrical engineering from Cooper Union in 1985 and his PhD degree in computer science from Columbia University in 1990. He is currently an associate professor of computer science at the City College of New York. His research interests include image processing, computer graphics, and computer vision. Prof. Wolberg is the recipient of a 1991 U.S. National Science Foundation Presidential Young Investigator Award and the 1997 CCNY Outstanding Teaching Award. He is the author of *Digital Image Warping* (IEEE CS Press, 1990), the first comprehensive monograph on warping and morphing.

Sung Yong Shin received his BS degree in industrial engineering in 1970 from Hanyang University, Seoul, Korea, and his MS and PhD degrees in industrial and operations engineering from the University of Michigan, Ann Arbor, in 1983 and 1986, respectively. He has been working at the Korea Advanced Institute of Science and Technology (KAIST) since 1987. He is a full professor in the Department of Computer Science at KAIST, Taejon, Korea. His research interests include computer graphics and computational geometry.