1995

Adaptive Reconstruction of Surfaces and Scalar Fields from Dense Scattered Trivariate Data

Chandrajit L. Bajaj

Fasuto Bernardini

Guoliang Xu

Report Number:
95-028

ADAPTIVE RECONSTRUCTION OF SURFACES AND SCALAR FIELDS FROM DENSE SCATTERED TRIVARIATE DATA

Chandrajit L. Bajaj
Fausto Bernardini
Guoliang Xu

CSD-TR-95-028
April 1995

Adaptive Reconstruction of Surfaces and Scalar Fields from Dense Scattered Trivariate Data *

Chandrajit L. Bajaj Fausto Bernardini † Guoliang Xu

Department of Computer Sciences
Purdue University
West Lafayette, Indiana

{bajaj,fxb,xuguo}@cs.purdue.edu, Tel: 317-494-6531

Abstract

In this paper, we propose an approach to model a set of trivariate dense and scattered data on an unknown manifold by the zero contour of the C^1 piecewise tri-quadratic or tri-cubic polynomial. The C^1 function approximates the data over an octree-like mesh in an adaptive fashion. At the same time, and applying the same techniques, a scalar field (scalar values associated with the sampled surface points) is also modeled as a C^1 piecewise polynomial function defined on the reconstructed surface. The reconstructed surface and field can be used to visualize and interact with the data in a variety of ways.

1 Introduction

In many important cases, a physical phenomenon is measured by sampling the value of a scalar or vector field at points on the surface of some object. For example, one might have sampled the temperature on the surface of a jet-engine, or the acceleration of a fluid flow on the wing of an airplane. Other cases of interest are electroencephalogram data on the surface of the scalp, or the amount of precipitation on the earth. In this paper we will consider the problem of reconstructing both the surface of an object and a (possibly multivariate) field on it from scattered, dense data (what we mean by *dense* will be more clearly stated later in this paper). We will assume that both the surface and the field are continuous and have continuous first-order derivatives. Since data measurements are subject to error, we will not try to exactly interpolate the data but rather approximate it within a given tolerance. We will define a way to measure the error-of-fit for both the surface and the data later in the paper. The problem may be formally stated as follows:

Definition 1.1 *Given a set of dense scattered points $P = \{p_i\}_{i=1}^N \subset \mathbf{R}^3$ on an unknown manifold M, and associated data values $V = \{v_i\}_{i=1}^N$, construct a smooth surface $S : s(x, y, z) = 0$, such that S approximates P within a given error bound ϵ_S, and a smooth function $F : f(x, y, z)$, such that F approximates the data V associated with P within a given error bound ϵ_F.*

*Supported in part by NSF grants CCR 92-22467, DMS 91-01424, AFOSR grants F49620-93-10138, F49620-94-1-0080, ONR grant N00014-94-1-0370 and NASA grant NAG-93-1-1473.

† Additional partial support from CNR, Italy.

The problem of reconstructing the approximation S to the unknown manifold M has attracted the interest of many authors. A number of methods have been developed for its solution [14, 5, 28, 9, 26, 7, 23, 20, 30, 17, 27, 8, 22, 2]

Most of the known methods use parametric or functional surface patches in either local interpolation or global interpolation. A few papers (see [29, 10, 18, 4, 11, 3, 2]) use implicit surface patches. In this paper, we use a piecewise implicitly defined tensor-product algebraic surface to approximate the unknown surface M.

The problem of interpolating data defined over a *given* manifold in $\mathbf{R}^3$ is commonly referred to as *modeling 3D scattered manifold data* or the *surface-on-surface* problem [24]. Prior work on this subject includes [5, 16, 1, 15, 25, 6]

Our method is based on constructing an approximation of the signed-distance function $\delta(p, M)$ defined in Section 2. A similar approach has been used in some of the papers cited above. A novelty of the method proposed in this paper is in the *adaptive* approximation of δ with tensor-product Bernstein-Bézier polynomial patches, with the required continuity conditions. Compared to the methods mentioned above, our approach has the advantage of using an octree-like cubic mesh for any scattered data. This makes it easy to use, and adaptively capable of handling irregular data. Moreover, it handles the problems of reconstructing the surface and r scalar fields on it with a uniform approach. Once a piecewise Bernstein-Bézier polynomials representation of the surface and the scalar fields has been constructed, the data can be easily visualized and interacted with. Polygonal shading or ray tracing can be used to display the reconstructed surfaces. Isocontouring, or the normal projection method can be used to display the fields over the surface. The weights (coefficients) of the various patches can be interactively and locally modified.

Paper overview: In Section 2 we define some concepts and give an outline of the algorithm. Section 3 describes how to define an approximate signed-distance function. We then illustrate in Section 4 how to adaptively approximate the signed-distance function with a piecewise polynomial. Some details on the approximation strategy are given in Section 5, while several examples are shown and discussed in Section 7. The Appendices contain a review of definitions related to Bernstein-Bézier polynomials and their properties, and the proof of some of the Lemmas and Theorems stated in the paper.

2 Outline of the algorithm

If M is a connected and orientable surface in $\mathbf{R}^3$, then it is possible to define (in all $\mathbf{R}^3$) a function $\delta(p)$, called signed-distance, by

$$\delta(p) := sign \cdot dist(p, M)$$

where $dist(p, M)$ denotes the Hausdorff distance from the point p to the surface M and the $sign$ is chosen so that $\delta(p)$ is positive when p is on one side of M, and negative when p lies on the other side. Then $\delta(p) = 0$ will recover the surface M.

When only discrete data on a surface is available, an approximate signed-distance can be defined in some appropriate way (see e.g. [23, 20, 2]). Given the signed-distance function, one can approximate it with a piecewise polynomial function $s(x, y, z)$ (in a suitable domain containing P), and then extract the zero-contour of s.

The algorithm proposed in this paper consists of the following phases:

1. **Build an approximation of the signed-distance function.** Preprocess the data so that, for a given query point q, the (approximate) value of $\delta(q, M)$ cam be computed. Notice that this requires a topologically consistent reconstruction of the surface orientation at each point. This step can be seen as transforming the problem from a surface-data reconstruction to a volume-data approximation.

2. **Approximate the signed-distance by a piecewise polynomial function.** Build, in an adaptive fashion, a piecewise polynomial, C^1-smooth approximation $s(x, y, z)$ of $\delta(p, M)$. The piecewise polynomial is built by least squares fitting of trivariate polynomials, in each cube of an octree-like subdivision of a domain containing P, to the data points within the cube and to additional samples of the signed-distance function δ defined in phase 1 above. If the error-of-fit in a cube of the subdivision exceeds the given bounds, then the cube is subdivided into eight sub-cubes and the process is repeated in each sub-cube. The reconstructed domain is implicitly defined as $s(x, y, z) = 0$.

3. **Approximate the scalar field defined over M.** Concurrently to the approximation of the signed-distance function, a piecewise polynomial approximation of the scalar field can be computed in a similar fashion by least squares fitting of the scalar field data in each cube in the octree.

In the following sections we will detail the algorithm outlined above.

3 From surface data to volume data: the signed-distance function

Using the signed-distance function to reconstruct a surface from scattered data points has been considered by several authors.

Moore and Warren [23] use a tetrahedral decomposition of the space, and reconstruct the surface by implicit barycentric Bernstein-Bézier patches. The signed-distance function used is sampled at data points (where it is obviously zero) and at *auxiliary* points, chosen as the vertices of a regular partitioning of each tetrahedron into sub-tetrahedra. If the sampling is dense enough, then the sub-tetrahedra containing data points partition the tetrahedron into two components, so that a sign can be associated with the distance at each vertex of the grid. This dense-sampling assumption can be too restrictive in some practical cases.

Hoppe et al. [20] use a more global approach to correctly orient the approximated manifold. First, for each data point p_i, they compute a best fit plane, and the associated normal $\hat{n}_i$, based on k neighboring points. Then they build the *Riemannian Graph*, $RG(P)$ over P (two points $p_i, p_j \in P$ are connected by an edge in $RG(P)$ iff either p_i is in the k-neighborhood of p_j or p_j is in the k-neighborhood of p_i). Each edge (i, j) is assigned the weight $1 - |\hat{n}_i \cdot \hat{n}_j|$, and a minimum spanning tree is computed. They then orient the plane associated with the point with the largest z-value so that its normal points toward the positive z-direction, and propagate this orientation to other points traversing the minimum spanning tree. The traversing order implicit in the MST avoids, in many examples illustrated in their paper, an incorrect orientation of parts of the manifold. Their method continues with the construction of a regular subdivision of a parallelepiped containing the data points into cubes. The value of δ is computed at all vertices of the subdivision as the signed distance of the vertex from the oriented plane associated with the closest point in P. An algorithm similar to *marching cubes* is then used to construct a piecewise-linear approximation of the zero contour of δ. In two subsequent steps, described in [21, 19], the constructed mesh is optimized (i.e., the number of triangles is reduced while the distance of the mesh from the data points is kept small) and then a smooth surface is built on it. While this approach gives very convincing results, and allows for smooth objects with sharp features to be correctly reconstructed, the computational time required by the optimization and smoothing steps is significant.

Bajaj et al. [2] propose the use of α-shapes [12, 13] to build a piecewise-linear approximation of the surface being reconstructed. The α-shape is a sub-complex of the Delaunay triangulation of the set of points P. This approach has the advantage of being based on a well-founded mathematical definition of the *shape* of a set of points. The piecewise-linear approximation is then used to compute the value of the signed-distance function at any point. A piecewise polynomial approximation is subsequently built on an adaptive Delaunay 3D triangulation of a domain containing P.

All the three schemes described above have advantages and disadvantages. In all three cases, the method used to define an approximate signed-distance function can be used as the first step of our algorithm. In our current implementation. we are using a *propagation* approach similar to that of [20]:

Begin Algorithm

1. **Local approximation.** Let P be the given surface data, then for each point $p \in P$, construct a local linear approximation $l_p(x, y, z) = ax + by + cz + d$ by least squares fitting p and other $k(\geq 2)$ nearest points of P. If the points used are collinear the number of points used is increased.

2. **Orienting normals.** For each point $p \in P$, construct a normal n_p from the local approximation $l_p(x, y, z) = 0$. That is, $sign \cdot \nabla l_p(p)/\|\nabla l_p(p)\|$. The direction (the $sign$ in the formula) of the normal must be chosen so that all the normals point toward the same side of the surface.

   In the following we assume that the surface data P is ρ-dense, that is any sphere with radius ρ and center in M contains at least one point in P. Our propagation algorithm starts with assigning an orientation to the six points having minimum or maximum x, y or z coordinate (the normal at the point with max z coordinate clearly must point upward, etc). Points whose associated normal has been oriented, but such that neighboring points might not have been oriented are called *boundary points* and are kept in a list. A point p is extracted from the list and all points contained in a ball centered in p and of radius ρ' (ρ' is a parameter to be chosen a priori, depending on the density ρ of the data set) are oriented (if they had not been oriented before) accordingly to the orientation of p (i.e., in such a way that the the scalar product of the two normals is positive).

3. **signed-distance function evaluation.** The distance $|\delta(p, M)|$ is computed by first finding a point $q \in P$ that has minimal distance to p. Then $|\delta(p, M)|$ is defined as the local minimal distance from p to $l_q(x, y, z) = 0$ around q. If the query point is outside the sphere $S(q, r)$, with center q and radius r, then use $|\delta(p, M)| = \|p - q\|$. The radius r can be taken to be the maximal distance of the k points that defined l_q from q. The sign of $|\delta(p, M)|$ is taken to be the sign of the inner product of n_q and $p - q$.

End Algorithm

4 Piecewise polynomial approximation of signed-distance

In this section, we describe how to use piecewise polynomials to approximate the signed-distance function. Let $D = [\alpha_1, \alpha_2] \times [\beta_1, \beta_2] \times [\gamma_1, \gamma_2]$ be a parallelepiped containing the data set P. The outline of the algorithm is as follows:

Begin Algorithm

1 **Initial Partition.** Construct a partition of D. That is choose a_i, b_j, c_k so that

$$\begin{array}{ccccccccc} \alpha_1 & = & a_0 & < & a_1 & < & \ldots & < & a_{l_1} & = & \alpha_2 \\ \beta_1 & = & b_0 & < & b_1 & < & \ldots & < & b_{l_2} & = & \beta_2 \\ \gamma_1 & = & c_0 & < & c_1 & < & \ldots & < & c_{l_3} & = & \gamma_2 \end{array}$$

and partition D by $D = \bigcup D_{ijk}$, with $D_{ijk} = [a_{i-1}, a_i] \times [b_{j-1}, b_j] \times [c_{k-1}, c_k]$ and $i = 1, \ldots, l_1, j = 1, \ldots, l_2, k = 1, \ldots, l_3$.

2 **Local Fitting.** For each element D_{ijk} that is within the distance ρ from P (if the cube is away from P by a distance bigger than ρ, we regard this cube as containing no surface), construct the (either tri-quadratic or tri-cubic) function $w = s_{ijk}(x, y, z)$ that fits the signed-distance function $\delta(p)$ at $p_s = (x_s, y_s, z_s) \in D_{ijk}$ in the least squares sense. The points p_s used are the points in $P \cap D_{ijk}$ and, additionally, the vertices of a regular grid on the cube D_{ijk} (the number of such points can be chosen depending on the number of data points available). The auxiliary grid points in the least square fit help preventing the function s_{ijk} from having multiple sheets.

3 **Adaptive Step.** Compute the algebraic distances of the computed patch from the data points. If the max distance is bigger than the given tolerance ϵ, subdivide D_{ijk} into eight equally-sized sub-cubes and return to Step 2. However, do not subdivide cubes whose side length is less than 2ρ.

End Algorithm

At the end of the iterative, adaptive fitting implemented by the steps outlined above, we have computed local polynomial approximations to the signed-distance function. Obviously, there is no guarantee that adjacent pieces join continuosly. Therefore we need an averaging phase (called free-form blending in [23]) in which values and derivatives at vertices of the octree subdivision are obtained evaluating the polynomials associated with incident cubes and taking their average (possibly weighted to take into account the number of data points used in the fitting and the goodness-of-fit achieved in each cube).

Notice that the adaptive refinement of the subdivision can be represented as an octree data structure. In the following we will refer to the *level* of a cube with the following meaning: cubes at level 1 are those with the coarser subdivision. Cubes at level $i + 1$ have been obtained subdividing a cube at a level i into eight sub-cubes.

Begin Algorithm

4 **Extract Values.** For each vertex of level 1 cubes in the final partition, compute function values and the required derivative values (C^3 data, see Section 5) by averaging the corresponding values of the neighboring cubes polynomials. Then compute a new polynomial in each level 1 cube using the averaged data.

5 **Recursive C^1 Interpolation.** Compute a new polynomial at each level 2 cube interpolating the C^3 data at its vertices. For vertices lying inside the face of an adjacent level 1 cube, first evaluate the polynomial in the adjacent cube to compute the C^3 data. Continue in a similar fashion with level i cubes, for $i > 2$, until all patches have been computed.

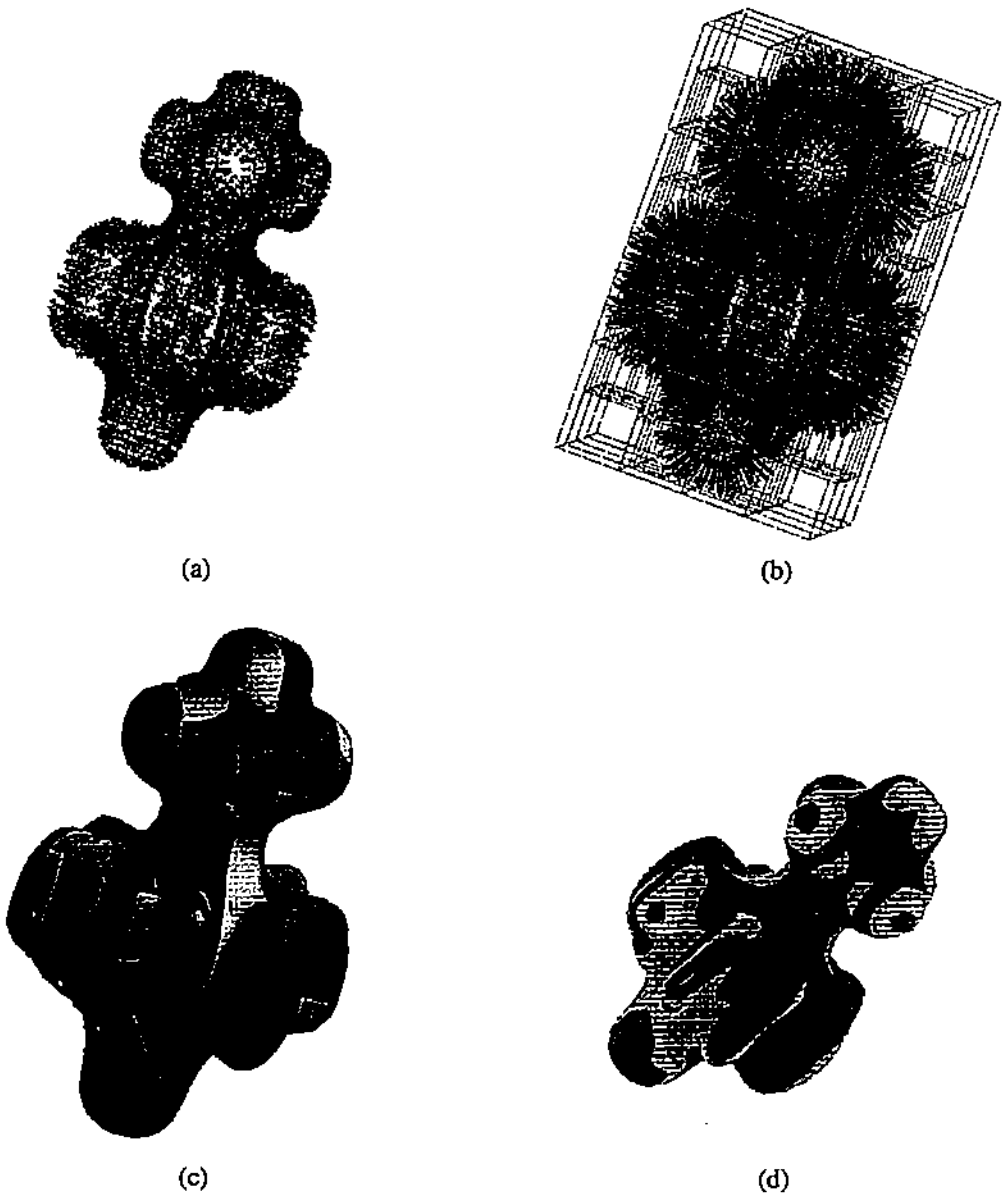

FIGURE 4.1: Reconstruction of a surface and an associated scalar field from scattered data: (a) Input points. (b) Orientation of normals and octree subdivision. (c) Piecewise polynomial approximation. (d) Reconstructed scalar field.

End Algorithm

At the end of the top-down construction of interpolants, we have obtained a C^1 smooth, piecewise trivariate polynomial, whose zero-contour approximates the points P.

Step 3 guarantees that the scheme is adaptive. The final partition produced by the algorithm is in general not uniform. Hence two adjacent cubes may have different sizes. Two adjacent cubes that share a part of the common face join in one of the following two fashions:

1. The two faces coincide completely.

2. One contains the other as a proper subset.

If the second case happens, then in Step 5 the interpolant for the larger cube will be computed first, and then this interpolant will be used to compute the C^3 data at the remaining vertices of the smaller cube. This guarantees the global C^1 continuity of the constructed function. We shall detail the construction method for the interpolants in the following sections.

In Step 2, we use algebraic polynomials to fit the data. Since the least squares approximation is not an exact fit, one might question if an approximate fit to the signed-distance could possibly lead to a multi-sheeted surface. For this, a theorem like Theorem 1 in [23] for a tetrahedron can be established for our cubic scheme:

Theorem 4.1 *Let $P_{ijk} = P \cap D_{ijk}$. For a sufficiently small ϵ, if there exists a plane $\pi(x,y,z) = 0$ such that all points of P_{ijk} lie within a distance ϵ from the plane, then the zero contour of the local fitting $s_{ijk}(x,y,z)$ is smooth, single sheeted in D_{ijk} and lies within some distance, depending only on ϵ and the degree of s_{ijk}, from the plane $\pi(x,y,z) = 0$.*

Proof: See Appendix (Theorem B.1). ◇

Another way of guaranteeing single-sheetedness is the following. In [26] is has been pointed out that if all the weights increase or decrease monotonically along one of the coordinate directions, then straight lines parallel to that direction intersect the surface patch at most once, i.e. the patch is single-sheeted. We state another characterization in the following:

Lemma 4.1 *If there exists an integer l $(0 < l < m)$ such that*

$$w_{ijk} \leq 0, \quad i = 0,1,...,l-1; \quad j = 0,1,...,n; \; k = 0,1,...,q \tag{4.1}$$

$$w_{ijk} \geq 0, \quad i = l+1,...,m; \quad j = 0,1,...,n; \; k = 0,1,...,q \tag{4.2}$$

and there is at least one strict inequality in each set of inequalities, then straight lines parallel to the x-direction intersect the surface patch exactly once. Similar conclusions hold for the y and z-direction intersections.

Proof: The Lemma can be easily proved using the variation-diminishing property of Bernstein-Bézier polynomials. ◇

Since the averaging in Step 4 will make the final approximation differ from the local approximation of Step 2, one should note that this change may destroy the smooth and single sheeted properties of the local approximation. To avoid this from happening, the averaged values should have a small difference from the local values. If the data is dense and the partition is fine enough, this will be guaranteed.

5 C^1 Interpolation of C^3 data by $(3,3,3)$- and $(2,2,2)$-polynomials

By C^3 data of a function f at a point p, we mean that we are given values at p for

$$f, \quad \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z}, \quad \frac{\partial^2 f}{\partial x \partial y}, \quad \frac{\partial^2 f}{\partial x \partial z}, \quad \frac{\partial^2 f}{\partial y \partial z}, \quad \frac{\partial^3 f}{\partial x \partial y \partial z}. \tag{5.1}$$

This Section shows a way of constructing tri-cubic and tri-quadratic interpolants over a cube with C^3 data on its vertices so that the composite function is C^1 continuous. The following Lemma tells us how to compute the BB form coefficients around a vertex from the C^3 data there.

Lemma 5.1 *Let* $W(x,y,z) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q} w_{ijk} B_i^m(u) B_j^n(v) B_k^q(w)$, $m > 0$, $n > 0$, $q > 0$, *be a BB form polynomial on the cube* $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$. *Then* W *interpolates the* C^3 *data (5.1) at the vertex* (a_1, b_1, c_1) *if and only if*

$$w_{000} = f \tag{5.2}$$

$$w_{100} = w_{000} + \frac{a_2 - a_1}{m} \frac{\partial f}{\partial x} \tag{5.3}$$

$$w_{010} = w_{000} + \frac{b_2 - b_1}{n} \frac{\partial f}{\partial y} \tag{5.4}$$

$$w_{001} = w_{000} + \frac{c_2 - c_1}{q} \frac{\partial f}{\partial z} \tag{5.5}$$

$$w_{110} = w_{010} + w_{100} - w_{000} + \frac{(a_2 - a_1)(b_2 - b_1)}{mn} \frac{\partial^2 f}{\partial x \partial y} \tag{5.6}$$

$$w_{101} = w_{001} + w_{100} - w_{000} + \frac{(a_2 - a_1)(c_2 - c_1)}{mq} \frac{\partial^2 f}{\partial x \partial z} \tag{5.7}$$

$$w_{011} = w_{001} + w_{010} - w_{000} + \frac{(b_2 - b_1)(c_2 - c_1)}{nq} \frac{\partial^2 f}{\partial y \partial z} \tag{5.8}$$

$$w_{111} = w_{011} + w_{101} - w_{001} + w_{110} - w_{010} - w_{100} + w_{000} + \frac{(a_2 - a_1)(b_2 - b_1)(c_2 - c_1)}{mnq} \frac{\partial^3 f}{\partial x \partial y \partial z} \tag{5.9}$$

Similar conclusions hold for the other 7 vertices of the cube D.

Proof: See Appendix (Lemma B.1). ◇

Lemma 5.2 *Let* W_1 *and* W_2 *be polynomials defined on equally sized cubes* D_1 *and* D_2, *adjacent along the* x*-direction (see A.5). Then if both* W_1 *and* W_2 *interpolate* C^3 *data at the four common vertices of* D_1, D_2, *four* x*-direction collinear conditions are satisfied at each of the common vertices. Similar conclusions hold for* y*- or* z*-adjacent cubes.*

Proof: See Appendix (Lemma B.2). ◇

Theorem 5.1 *If* W_1 *and* W_2 *in Lemma above are tri-cubic, and if both* W_1 *and* W_2 *interpolate* C^3 *data at the common four vertices of two adjacent cubes* D_1 *and* D_2, *then* W_1 *and* W_2 *are* C^1 *continuous on the common face of* D_1 *and* D_2.

Proof: See Appendix (Theorem B.2). ◇

The last Theorem says that, if a volume consists of cubes, and if a BB polynomial on each cube is constructed from C^3 data at the vertex by formulae (5.2)—(5.9), then the composite function is C^1 continuous. The discussion above also shows that $m = n = q = 3$ is the minimal degree for forming C^1 piecewise functions, since there is no degree of freedom left. If a lower degree polynomial is used or some degrees of freedom are required, one has to subdivide each cube into smaller sub-cubes.

Creating degrees of freedom by using tri-cubic interpolation. As mentioned in Section 4, the purpose of creating degrees of freedom is to achieve better approximation to the local fitting. For a given cube D with C^3 data on its eight vertices, the tri-cubic interpolating W_D is uniquely defined as described above. To create some degrees of freedom, we subdivide the cube D into eight equally-sized sub-cubes. At each center of the faces of D, we determine the C^3 data by evaluating W_D and its derivatives. The C^3 data at the center c of the cube D is free. That is, eight degrees of freedom are created by this subdivision. This C^3 data can be obtained by evaluating the local fitting at c, or by interpolation as shown in the following. Let $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$. Consider an interpolant W_{D_1} over $D_1 = [\frac{a_1+a_2}{2}, a_2] \times [\frac{b_1+b_2}{2}, b_2] \times [\frac{c_1+c_2}{2}, c_2]$. Except for c, the cube D_1 has C^3 data at seven of its vertices. Now we determine W_{D_1} by interpolating this C^3 data and the local fitting function at the following eight points

$$c,\ c + \Delta x,\ c + \Delta y,\ c + \Delta z,\ c + \Delta x + \Delta y,\ c + \Delta x + \Delta z,\ c + \Delta y + \Delta z,\ c + \Delta x + \Delta y + \Delta z$$

where $\Delta x = \frac{a_2-a_1}{4}, \Delta y = \frac{b_2-b_1}{4}, \Delta x = \frac{c_2-c_1}{4}$. Then we determine the C^3 data at c by evaluating W_{D_1} and its derivatives at c. Now the other seven interpolants over the sub-cubes are ready to be defined. Of course, this subdivision process can be repeated if necessary. The composite function is always C^1 continuous. Furthermore, the C^3 data at the center of the face of D can also be determined by interpolating the local fitting. However, this will affect the interpolant of the neighboring cube.

Tri-quadratic interpolation. C^1 interpolation of the C^3 data can also be achieved by a tri-quadratic piecewise polynomial. However, this requires splitting each cube in eight sub-cubes to create additional degrees of freedom in the choice of coefficients.

The first Lemma in the following says that the six C^1 conditions on a plane and around a vertex are not independent.

Lemma 5.3 *Let $R = [a_1, a_3] \times [b_1, b_3]$ be a rectangle in the plane (see Figure 5.1(a)), and $w_1, \ldots, w_9$ be values on the nine grid points. Then the six collinear conditions*

$$\frac{w_{i+1} - w_i}{a_2 - a_1} = \frac{w_{i+2} - w_{i+1}}{a_3 - a_2}, \quad i = 1, 4, 7 \tag{5.10}$$

$$\frac{w_{i+3} - w_i}{b_2 - b_1} = \frac{w_{i+6} - w_{i+3}}{b_3 - b_2}, \quad i = 1, 2, 3 \tag{5.11}$$

are satisfied if any five of them are satisfied.

Proof: See Appendix (Lemma B.3). ◇

The next Lemma guarantees that by subdivision, the tri-quadratic interpolants over the eight sub-cubes exist uniquely and the composite function is C^1 continuous.

Lemma 5.4 *Let $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$ be a given cube. At each of its eight vertices $P_{i_1 i_2 i_3}$, we are given C^3 data. If we subdivide D into eight sub-cubes $D_{i_1 i_2 i_3}$ (see Figure 5.1(b)), then there exists uniquely one piecewise function W on D such that*

a. $W_{i_1 i_2 i_3} = W|_{D_{i_1 i_2 i_3}}$ is a (2,2,2)-polynomial, that interpolates the set of C^3 data at $P_{i_1 i_2 i_3}$.

b. W is C^1 continuous on D.

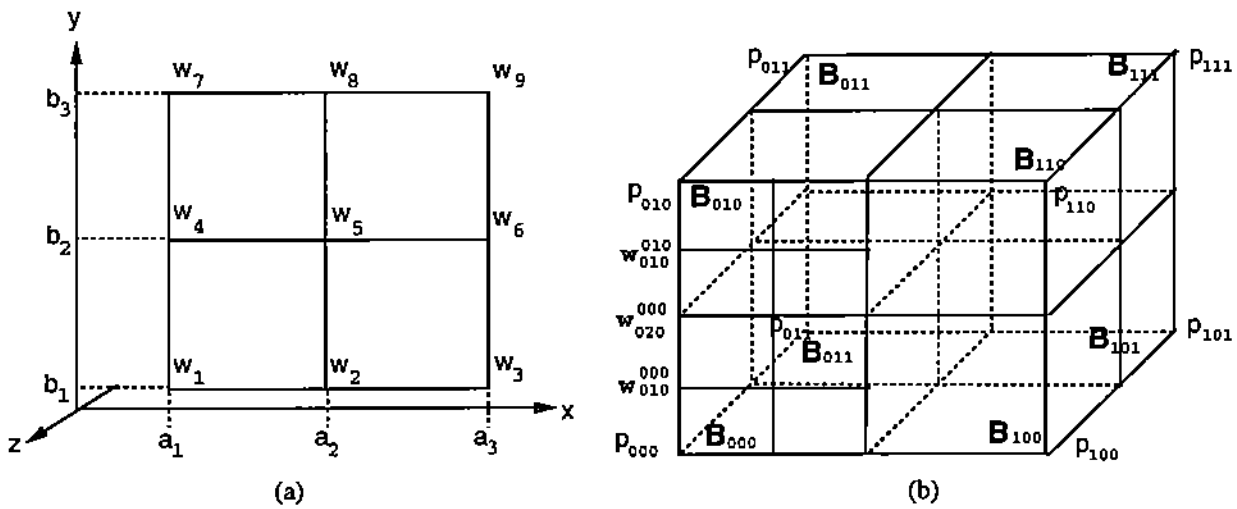

FIGURE 5.1: (a) Six collinear conditions. (b) Subdivision of a cube D into eight sub-cubes.

c. *If W' is defined in the same way over an adjacent cube D', then W and W' are C^1 continuous on the common face of D and D'.*

Proof: See Appendix (Lemma B.4). ◇

Unlike the tri-cubic interpolation, the subdivision does not lead to extra degrees of freedom.

6 C^1 Interpolation of C^1 data by (2,2,2) and (3,3,3) polynomials

From the discussion of Section 5, we know that if eight Bézier coefficients around one vertex are determined in such a way that collinear conditions are satisfied, then a globally C^1 interpolant to C^3 data can be constructed using either tri-cubic or tri-quadratic (with subdivision into sub-cubes) polynomials In this section, we use C^1 data instead of C^3 data to determine the required eight coefficients around each corner of the cube. If these coefficients are determined in a way such that they satisfy the collinear conditions with its neighboring cubes coefficients, then the global C^1 continuity is achieved as before. The C^1 data to be interpolated is f, $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial z}$.

From (5.2)–(5.5), the C^1 data at a vertex will determine the x, y and z direction coefficients, that is, w_{000}, w_{100}, w_{010} and w_{001}, around the vertex. The remaining coefficients at the vertex, that is, w_{110}, w_{101}, w_{011} and w_{111} have to be determined by the C^1 continuity conditions. The following two Lemmas study the solvability of these equations and tell us how many degrees of freedom we could have.

Lemma 6.1 *Let a, b, c, d and w be given (see Figure 6.1(a)) such that*

$$\frac{w-a}{\Delta x_1} = \frac{b-w}{\Delta x_2}, \quad \frac{w-c}{\Delta y_1} = \frac{d-w}{\Delta y_2} \tag{6.1}$$

Then there exist $w_1 \ldots w_4$ satisfying the equations

$$\frac{d-w_3}{\Delta x_1} = \frac{w_4-d}{\Delta x_2}, \quad \frac{c-w_1}{\Delta x_1} = \frac{w_2-c}{\Delta x_2}, \quad \frac{a-w_1}{\Delta y_1} = \frac{w_3-a}{\Delta y_2}, \quad \frac{b-w_1}{\Delta y_1} = \frac{w_4-b}{\Delta y_2} \tag{6.2}$$

and the solution can be expressed as

$$w_2 = \frac{\Delta x_1 + \Delta x_2}{\Delta x_1} c - \frac{\Delta x_2}{\Delta x_1} w_1 \tag{6.3}$$

$$w_3 = \frac{\Delta y_1 + \Delta y_2}{\Delta y_1} a - \frac{\Delta y_2}{\Delta y_1} w_1 \tag{6.4}$$

$$w_4 = \frac{\Delta x_1 + \Delta x_2}{\Delta x_1} d - \frac{\Delta x_2}{\Delta x_1}\frac{\Delta y_1 + \Delta y_2}{\Delta y_1} a + \frac{\Delta x_2 \Delta y_2}{\Delta x_1 \Delta y_1} w_1. \tag{6.5}$$

Proof: See Appendix (Lemma B.5). ◇

The Lemma says that the solution of equations (6.2) always exists and is one dimensional. In the following, we call equations (6.1) and (6.2) *2D-collinear conditions.*

Lemma 6.2 *For a given cube, let $(q_1, z_1, q_2, y_2, q_3, z_2, q_4, y_1, w)$, $(r_1, z_1, r_2, x_2, r_3, z_2, r_4, x_1, w)$ and $(s_1, x_1, s_2, y_2, s_3, x_2, s_4, y_1, w)$ satisfy the 2D-collinear conditions (see Figure 6.1(b)). Then there exist $w_1, \ldots, w_8$ satisfying twelve collinear conditions (each one corresponds to one edge of the cube), and the solution is one dimensional. More precisely, the set of twelve equations is equivalent to a subset of seven equations (E.g., four equations on a face and one from the opposite face can be removed).*

Proof: See Appendix (Lemma B.6). ◇

For to the cube in Figure 6.1(b) there are 27 collinear conditions (we will call them 3D-collinear conditions). Suppose $w, x_1, x_2, y_1, y_2, z_1 z_2$ are given, that is, these values are determined by the C^1 data: Then there are 20 unknowns. To satisfy the 3D-collinear condition, we have four degrees of freedom. For each of the three 2D-collinear condition given at the beginning of Lemma 6.2, we have one degree of freedom (see Lemma 6.1). For the 3D-collinear condition, we have another degree of freedom. Now we show how to use the four degrees of freedom:

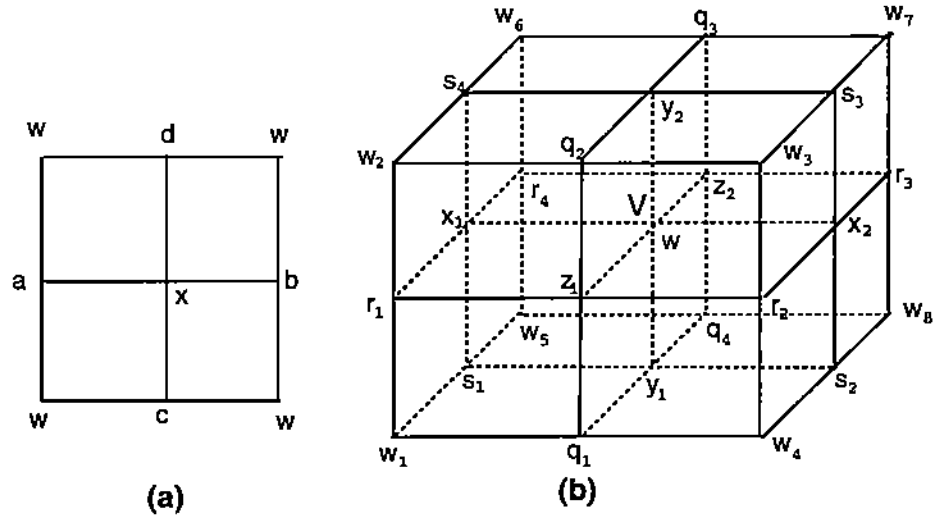

FIGURE 6.1: (a). 2D-collinear Conditions; (b). 3D-collinear Conditions

a. **2D-collinear condition d.o.f.** (see Lemma 6.1). Under the 2D-collinear condition (6.1) and (6.2), we choose w_i such that $w_i = w_i'$, $i = 1, 2, 3, 4$, in the least squares sense, where w_i' are the given values. We provide these values by evaluating the local fitting at the corresponding points. Therefore by (6.3)—(6.5) we can establish an equation in the form $\begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}^T w_1 = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix}^T$. The least squares solution is $w_1 = \sum_{i=1}^{4} v_i b_i / \sum_{i=1}^{4} v_i^2$. Substituting this into (6.3)—(6.5), we get w_2—w_4.

b. **3D-collinear condition d.o.f.** (see Lemma 6.2). Suppose $q_i, r_i, s_i (i = 1, 2, 3, 4)$ are determined, then under the 3D-collinear condition, we force $w_i = w_i'$, $i = 1, \ldots, 8$. Again, these w_i' are provided by evaluating the local fitting at the corresponding points. $w_2, \ldots, w_8$ are determined in a similar fashion.

C^1 Interpolation by C^1 Data Suppose we are given the C^1 data at each vertex of the cube. Now we show how to use Lemmas 6.1 and 6.2 to determine the eight coefficients around each vertex.

First, w_1, w_2, w_3, w_4 are determined from C^1 data at the vertex by using formulas (5.2)–(5.5). In order to determine the other four coefficients $w_5, \ldots, w_8$, we need to consider the 2D-collinear and 3D-collinear conditions for the seven neighboring cubes. Consider Figure 6.1(b). Suppose V is the common vertex of eight cubes (see Figure 6.1(b), where the eight cubes drawn are in fact the corner parts of the eight cubes). w is determined by f; x_1, x_2 are determined by $\frac{\partial f}{\partial x}$; y_1, y_2 are determined by $\frac{\partial f}{\partial y}$, and so do z_1 and z_2. Then by the discussion above, all other coefficients are determined. That is, $q_i (i = 1, \ldots, 4)$ are determined by the 2D-collinear condition and force $q_i = q_i'$ in the least square sense. The value q_i' can be obtained by averaging the values of the local approximation at the corresponding points. $r_i, s_i (i = 1, \ldots, 4)$ are similarly defined. So do the $w_1, \ldots, w_8$ by choosing $w_1' \ldots, w_8'$ as the values of the local approximation.

In this way, all the coefficients around the vertex V are determined and they satisfy the 3D-collinear condition. For each single cube that shares the vertex V, eight coefficients around V are determined. Knowing how to obtain the eight coefficients around the vertex, C^1 (2, 2, 2) (with subdivision) and (3, 3, 3)(without subdivision) interpolant can be built as in Section 5.

7 Examples of visualization

In this Section we present an example of a reconstructed object and an associated scalar field. The sampled points come from the surface of a jet engine, while the associated values measure the pressure on the engine (data from a

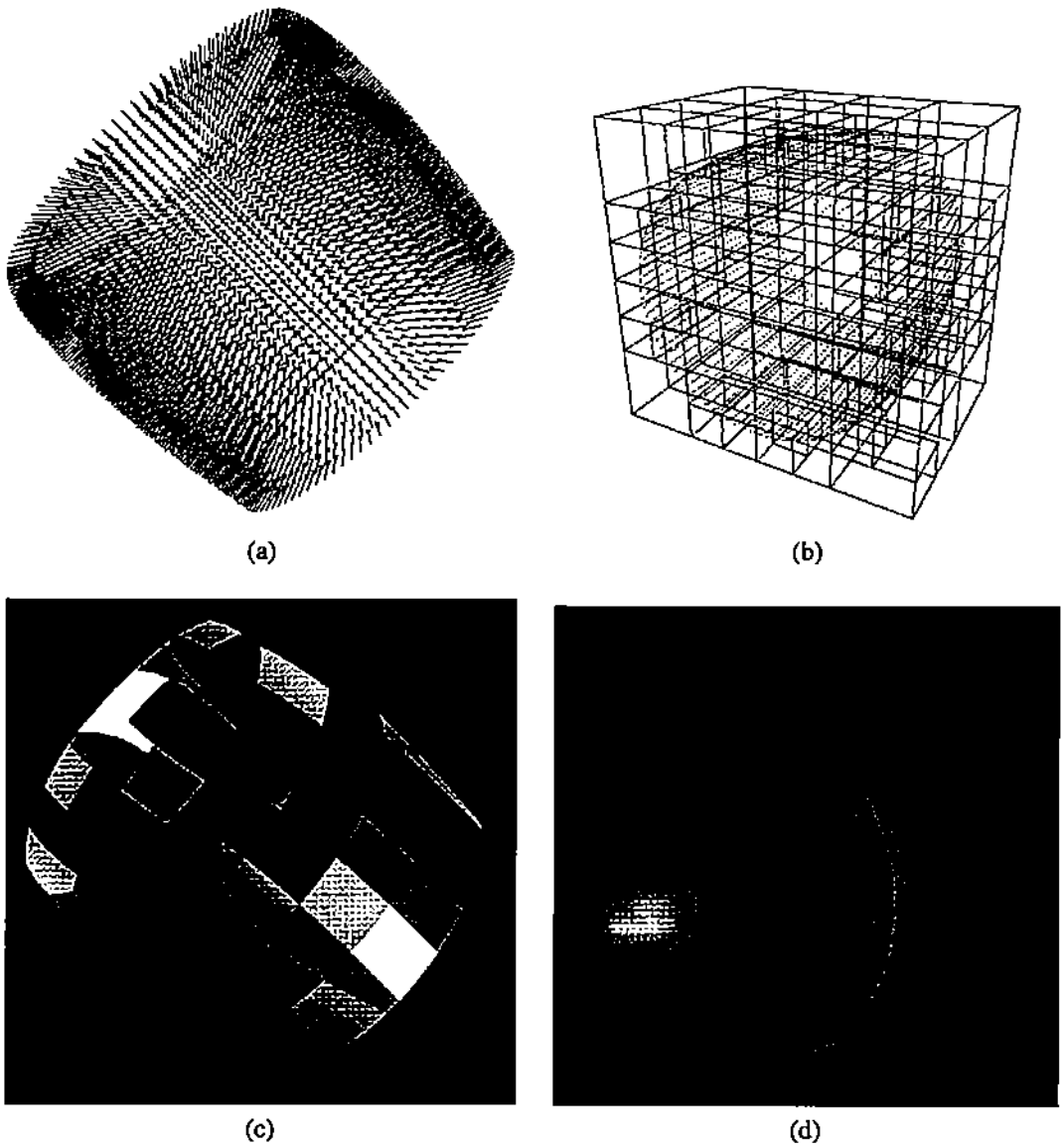

FIGURE 7.1: Reconstruction of a jet engine: (a) Input data for the outer cowl, with the oriented normals. (b) Octree subdivision generated by the approximation algorithm. (c) Piecewise polynomial approximation. (d) Reconstructed engine.

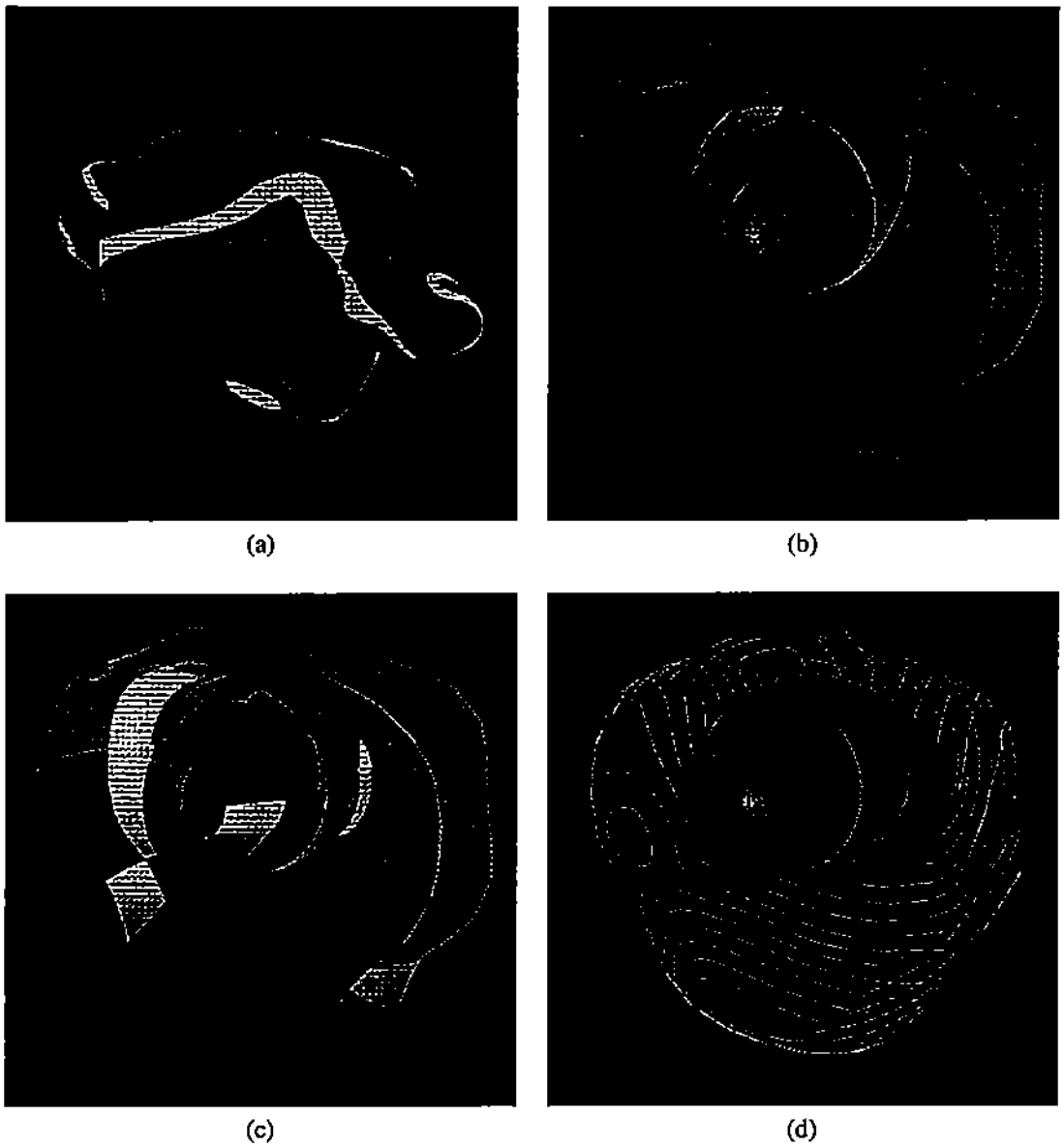

FIGURE 7.2: Visualization of the reconstructed jet engine and a pressure field on its surface: (a) Iso regions of the pressure field. (b) The pressure field displayed with the normal projection method (surface on surface). (c) The piecewise polynomial patches. (d) Isocontours of the pressure field displayed on the projected surface.

simulation).

Figure 7.1 shows several steps of the reconstruction process on one part of the engine. The 3780 data points for the outer cowl are preprocessed to associate local fitting planes and orient the associated normals (7.1(a)). The approximation algorithm begins with a given grid (in this case, a uniform subdivision into $5 \times 5 \times 5$ equally-sized cubes) and then adaptively refines it until the error bound conditions are met (the error in this example was set to 0.01 times the max size of the object). The final subdivision is displayed in Figure 7.1(b). After averaging, a C^1-smooth piecewise polynomial surface is obtained, as shown in Figure 7.1(c). The complete reconstruction of this object took about 30 seconds on a SGI MIPS4400 workstation. The full reconstructed engine is finally shown in Figure 7.1(d). At the same time, a different piecewise polynomial, whose domain is the same as for the surface implicit function, approximates the sampled scalar field (see Figure 7.2).

Figure 7.2 shows four different visualization of the jet engine data. In Figure 7.2(a) some isoregions of the pressure field have been drawn on the engine surface. In Figure 7.2(b) we have used the normal-projection method to show the scalar field as a surface-on-surface: Points on the domain surface have been projected along the surface normal direction to a distance proportional to the value of the field at that point. The field surface patches are visible in Figure 7.2(c). Finally, in Figure 7.2(d), we show isocontours projected on the surface-on-surface.

References

[1] ALFELD, P. Scattered data interpolation in three or more variables. In *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker, Eds. Academic Press, 1989, pp. 1–34.

[2] BAJAJ, C., BERNARDINI, F., AND XU, G. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Computer Graphics Proceedings, Annual Conference Series* (1995), Proceedings of SIGGRAPH 95, ACM SIGGRAPH. To appear.

[3] BAJAJ, C., CHEN, J., AND XU, G. Modeling with cubic A-patches. *ACM Transactions on Graphics* (1995). To Appear.

[4] BAJAJ, C., AND IHM, I. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics 19*, 1 (Jan. 1992), 61–91.

[5] BARNHILL, R. Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design 2* (1985), 1–17.

[6] BARNHILL, R. E., OPITZ, K., AND POTTMANN, H. Fat surfaces: a trivariate approach to triangle-based interpolation on surfaces. *Computer Aided Geometric Design 9* (1992), 365–378.

[7] BOLLE, R. M., AND VEMURI, B. C. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 1 (Jan. 1991), 1–13.

[8] CHEN, X., AND SCHMITT, F. Surface modelling of range data by constrained triangulation. *Computer Aided Design 26*, 8 (Aug. 1994), 632–645.

[9] CHOI, B. K., SHIN, H. Y., YOON, Y. I., AND LEE, J. W. Triangulation of scattered data in 3D space. *Computer Aided Design 20*, 5 (June 1988), 239–248.

[10] DAHMEN, W. Smooth piecewise quadratic surfaces. In *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker, Eds. Academic Press, Boston, 1989, pp. 181–193.

[11] DAHMEN, W., AND THAMM-SCHAAR, T.-M. Cubicoids: modeling and visualization. *Computer Aided Geometric Design 10* (1993), 93–108.

[12] EDELSBRUNNER, H., KIRKPATRICK, D., AND SEIDEL, R. On the shape of a set of points in the plane. *IEEE Trans. on Information Theory 29*, 4 (1983), 551–559.

[13] EDELSBRUNNER, H., AND MUCKE, E. P. Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (Jan. 1994), 43–72.

[14] FAUGERAS, O. D., HEBERT, M., MUSSI, P., AND BOISSONNAT, J. D. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics and Image Processing 25* (1984), 169–183.

[15] FOLEY, T. A. Interpolation to scattered data on a spherical domain. In *Algorithms fo Approximation II*, M. Cox and J. Mason, Eds. Chapman and Hall, London, 1990, pp. 303–310.

[16] FRANKE, R. Recent advances in the approximation of surfaces from scattered data. In *Multivariate Approximation*, C.K.Chui, L.L.Schumarker, and F.I.Utreras, Eds. Academic Press, New York, 1987, pp. 275–335.

[17] GOSHTASBY, A. Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces. *International Journal of Computer Vision 10*, 3 (1993), 233–256.

[18] GUO, B. *Modeling arbitrary smooth objects with algebraic surfaces.* PhD thesis, Computer Science, Cornell University, 1991.

[19] HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWITZER, J., AND STUELZLE, W. Piecewise smooth surface reconstruction. In *Computer Graphics Proceedings, Annual Conference Series* (1994), ACM SIGGRAPH, pp. 295–302.

[20] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUELZLE, W. Surface reconstruction from unorganized points. *Computer Graphics (Proc. of the SIGGRAPH Conf.) 26*, 2 (July 1992), 71–78.

[21] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUELZLE, W. Mesh optimization. In *Computer Graphics Proceedings, Annual Conference Series* (1993), ACM SIGGRAPH, pp. 19–26.

[22] LIAO, C. W., AND MEDIONI, G. Surface approximation of a cloud of 3D points. *Graphical Models and Image Processing 57*, 1 (Jan. 1995), 67–74.

[23] MOORE, D., AND WARREN, J. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the twenty-fourth annual Hawaii International Conference on System Sciences* (1991), V. Milutinovic and B. D. Shriver, Eds., vol. 1.

[24] NIELSON, G. M. Scattered data modeling. *IEEE Computer Graphics & Applications 13* (1993), 60–70.

[25] NIELSON, G. M., FOLEY, T. A., HAMANN, B., AND LANE, D. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics & Applications 11*, 3 (May 1991), 47–55.

[26] PATRIKALAKIS, N. M., AND KRIEZIS, G. A. Representation of piecewise continuous algebraic surfaces in terms of B-splines. *The Visual Computer 5* (1989), 360–374.

[27] POLI, R., COPPINI, G., AND VALLI, G. Recovery of 3D closed surfaces from sparse data. *Computer Vision, Graphics and Image Processing 60*, 1 (July 1994), 1–25.

[28] SCHMITT, F., BARSKY, B. A., AND DU, W. An adaptive subdivision method for surface fitting from sampled data. *Computer Graphics (Proc. of the SIGGRAPH Conf.) 20*, 4 (1986), 179–188.

[29] SEDERBERG, T. W. Piecewise algebraic surface patches. *Computer Aided Geometric Design 2* (1985), 53–59.

[30] VELTKAMP, R. *Closed object boundaries from scattered points.* PhD thesis, Erasmus Universiteit Rotterdam, The Netherlands, 1992.

A Tensor-product Bernstein-Bézier form

Bernstein–Bézier (BB) form. For $(x, y, z) \in D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$, the BB form is

$$W(x, y, z) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q} w_{ijk} B_i^m(u(x)) B_j^n(v(y)) B_k^q(w(z)) \tag{A.1}$$

where $B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$,, $u(x) = \frac{x-a_1}{a_2-a_1}$, $v(y) = \frac{y-b_1}{b_2-b_1}$ and $w(z) = \frac{z-c_1}{c_2-c_1}$.

Partial derivatives of BB form.

$$\frac{\partial W}{\partial x} = \frac{m}{a_2 - a_1} \sum_{i=0}^{m-1} \sum_{j=0}^{n} \sum_{k=0}^{q} \Delta^{100} w_{ijk} B_i^{m-1}(u) B_j^n(v) B_k^q(w) \tag{A.2}$$

$$\frac{\partial W}{\partial y} = \frac{n}{b_2 - b_1} \sum_{i=0}^{m} \sum_{j=0}^{n-1} \sum_{k=0}^{q} \Delta^{010} w_{ijk} B_i^m(u) B_j^{n-1}(v) B_k^q(w) \tag{A.3}$$

$$\frac{\partial W}{\partial z} = \frac{q}{c_2 - c_1} \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q-1} \Delta^{001} w_{ijk} B_i^m(u) B_j^n(v) B_k^{q-1}(w) \tag{A.4}$$

where $\Delta^{100} w_{ijk} = w_{i+1,j,k} - w_{ijk}$, $\Delta^{010} w_{ijk} = w_{i,j+1,k} - w_{ijk}$, $\Delta^{001} w_{ijk} = w_{i,j,k+1} - w_{ijk}$.

Continuity. Let

$$W_\ell(x, y, z) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q} w_{ijk}^{(\ell)} B_i^m(u^{(\ell)}) B_j^n(v) B_k^q(w), \quad \ell = 1, 2 \tag{A.5}$$

be two BB form on the two cubes

$$D_\ell = [a_\ell, a_{\ell+1}] \times [b_1, b_2] \times [c_1, c_2], \quad \ell = 1, 2 \tag{A.6}$$

respectively, $u^{(\ell)} = \dfrac{x - a_\ell}{a_{\ell+1} - a_\ell}$. Then W_1 and W_2 are continuous at the common face $x = a_2$ iff

$$w_{mjk}^{(1)} = w_{0jk}^{(2)}, \quad j = 0, 1, \ldots, n, \quad k = 0, 1, \ldots, q \tag{A.7}$$

Smoothness. W_1 and W_2 defined in (A.5) are C^1 continuous at $x = a_2$ iff (A.7) holds and

$$\frac{w_{mjk}^{(1)} - w_{m-1jk}^{(1)}}{\Delta x_1} = \frac{w_{1jk}^{(2)} - w_{0jk}^{(2)}}{\Delta x_2}, \quad j = 0, 1, \ldots, n, \quad k = 0, 1, \ldots, q \tag{A.8}$$

where $\Delta x_1 = a_2 - a_1$, $\Delta x_2 = a_3 - a_2$, or

$$w_{mjk}^{(1)} = \frac{\Delta x_2 w_{m-1,jk}^{(1)}}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1 w_{1jk}^{(2)}}{\Delta x_1 + \Delta x_2}. \tag{A.9}$$

Note:

1. When we consider the C^1 continuity at the common face $x = a_2$, C^0 continuity guarantees that $\frac{\partial W_1}{\partial y} = \frac{\partial W_2}{\partial y}$, $\frac{\partial W_1}{\partial z} = \frac{\partial W_2}{\partial z}$. Then we need only to consider the cross derivative, i.e., $\frac{\partial W_\ell}{\partial z}$.
2. The cubes D_1 and D_2 defined in (A.6) are called x-adjacent. Similar conclusions hold for the functions defined on the y-adjacent and z-adjacent cubes.
3. (A.9) follows form (A.8) directly. Both of them are called collinear conditions. More specifically, we call them x-direction collinear conditions. Of course, y-direction and z-direction collinear conditions are also used.

B Proofs of Lemmas and Theorems

All the proofs of Lemmas and Theorems stated in the papers have been grouped in this appendix.

Theorem B.1 *Let $P_{ijk} = P \cap D_{ijk}$. For sufficiently small ϵ, if there exists a plane $\pi(x, y, z) = 0$ such that all points of P_{ijk} lie within a distance ϵ from the plane, then the zero contour of the local fitting $s_{ijk}(x, y, z)$ is smooth, single sheeted in D_{ijk} and lies within some distance, depending only on ϵ and the degree of s_{ijk}, from the plane $pi(x, y, z) = 0$.*

Proof: Let $\pi(x,y,z) = ax + by + cz + d$ with $a^2 + b^2 + c^2 = 1$. Then for any $p \in \mathbf{R}^3$, the distance from p to $\pi = 0$ is $|\pi(p)|$. Orient $\pi = 0$ properly so that $\pi(p)s_{ijk}(p) > 0$ for p that is away from the plane by a distance at least ϵ. Then for any p, $s_{ijk}(p) = \pi(p) + \delta(p)$ with $|\delta(p)| \le c\epsilon$, c is a constant. Let B be the coefficient vector of BB form of the local fitting s_{ijk}. That is, it satisfies the least squares system $A^TAx = A^TF$ with $F = [f(p_1), ...]^T$, where p_s are the interpolation points. Then B can be written as $B = B^l + B^\epsilon$, where

$$A^TAB^l = A^TL, \quad A^TAB^\epsilon = A^T\Delta$$

with $L = [\pi(p_1), ...]^T$, $\Delta = [\delta(p_1), ...]^T$. Hence

$$\|B^\epsilon\| \le \|(A^TA)^{-1}\| \, \|A^T\Delta\| \le C\epsilon$$

where C is a constant. WLG, assume $|a| = \max\{|a|, |b|, |c|\}$, that is $\pi(x,y,z)$ has the biggest x-directional derivative among the three x, y, z-directional derivatives. Since the polynomial with coefficient vector B^l is the same as $\pi(x,y,z)$, the coefficients b^l_λ in B^l is strictly monotonic in the x-direction. Therefore, we can choose ϵ so that

$$\|B^\epsilon\| \le |b^l_\lambda - b^l_{\lambda - e_1}| \tag{B.1}$$

for any index λ. That is, the coefficients b_λ in $B = B^l + B^\epsilon$ is strictly monotonic in the x-direction. Hence any straight line parallel to the x-axis will intersect the zero contour at most once. Hence the contour is smooth and single sheeted. Finally, it follows from (B.1) that the surface $s_{ijk} = 0$ must lie within the distance $|b^l_\lambda - b^l_{\lambda - e_1}|$ of $\pi = 0$. ◇

Lemma B.1 *Let $W(x,y,z) = \sum_{i=0}^m \sum_{j=0}^n \sum_{k=0}^q w_{ijk} B^m_i(u) B^n_j(v) B^q_k(w)$, $m > 0$, $n > 0$, $q > 0$, be a BB form polynomial on the cube $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$. Then W interpolates the C^3 data (5.1) at the vertex (a_1, b_1, c_1) if and only if*

$$w_{000} = f \tag{B.2}$$

$$w_{100} = w_{000} + \frac{a_2 - a_1}{m} \frac{\partial f}{\partial x} \tag{B.3}$$

$$w_{010} = w_{000} + \frac{b_2 - b_1}{n} \frac{\partial f}{\partial y} \tag{B.4}$$

$$w_{001} = w_{000} + \frac{c_2 - c_1}{q} \frac{\partial f}{\partial z} \tag{B.5}$$

$$w_{110} = w_{010} + w_{100} - w_{000} + \frac{(a_2 - a_1)(b_2 - b_1)}{mn} \frac{\partial^2 f}{\partial x \partial y} \tag{B.6}$$

$$w_{101} = w_{001} + w_{100} - w_{000} + \frac{(a_2 - a_1)(c_2 - c_1)}{mq} \frac{\partial^2 f}{\partial x \partial z} \tag{B.7}$$

$$w_{011} = w_{001} + w_{010} - w_{000} + \frac{(b_2 - b_1)(c_2 - c_1)}{nq} \frac{\partial^2 f}{\partial y \partial z} \tag{B.8}$$

$$w_{111} = w_{011} + w_{101} - w_{001} + w_{110} - w_{010} - w_{100} + w_{000} + \frac{(a_2 - a_1)(b_2 - b_1)(c_2 - c_1)}{mnq} \frac{\partial^3 f}{\partial x \partial y \partial z} \tag{B.9}$$

Similar conclusions hold for the other 7 vertices of the cube D.

Proof: Suppose W interpolates the C^3 data (5.1) at (a_1, b_1, c_1). By noting

$$(B^m_i B^n_j B^q_k)(a_1, b_1, c_1) = \begin{cases} 1 & i = j = k = 0 \\ 0 & \text{otherwise,} \end{cases}$$

(B.2) holds obviously. (B.3)—(B.5) follow directly from (A.2) and (A.4). Using formula (A.2)—(A.4) repeatedly, we get (B.6)—(B.9). The reverse side of the Lemma is similarly proved. ◇

Lemma B.2 *Let W_1 and W_2 be polynomials defined on equally sized cubes D_1 and D_2, adjacent along the x-direction (see A.5). Then if both W_1 and W_2 interpolate C^3 data at the four common vertices of D_1, D_2, four x-direction collinear conditions are satisfied at each of the common vertices. Similar conclusions hold for y- or z-adjacent cubes.*

Proof: Let W_1 and W_2 be defined as in (A.5). Consider the vertex (a_2, b_1, c_1). From (B.2), (B.4), (B.5) and (B.8), it follows that

$$w^{(1)}_{mij} = w^{(2)}_{0ij}, \quad i = 0, 1, \quad j = 0, 1.$$

From (B.3)

$$w^{(1)}_{m-1\,00} = w^{(1)}_{m00} + \frac{a_2 - a_1}{m}\frac{\partial f}{\partial x}.$$

Hence

$$\frac{w^{(1)}_{m-1,00} - w^{(1)}_{m00}}{a_2 - a_1} = \frac{w^{(2)}_{100} - w^{(2)}_{000}}{a_3 - a_2}. \tag{B.10}$$

Using (B.6) for W_1 and W_2 we have

$$\begin{aligned} w^{(1)}_{m-1,10} &= w^{(1)}_{m10} + w^{(1)}_{m-1,00} - w^{(1)}_{m00} + \tfrac{(a_2-a_1)(b_2-b_1)}{mn}\tfrac{\partial^2 f}{\partial x \partial y} \\ w^{(2)}_{110} &= w^{(2)}_{010} + w^{(2)}_{100} - w^{(2)}_{000} + \tfrac{(a_3-a_2)(b_2-b_1)}{mn}\tfrac{\partial^2 f}{\partial x \partial y} \end{aligned}$$

and hence

$$\frac{w^{(1)}_{m-1,10} - w^{(1)}_{m10}}{a_2 - a_1} + \frac{w^{(1)}_{m00} - w^{(1)}_{m-1,00}}{a_2 - a_1} = \frac{w^{(2)}_{110} - w^{(2)}_{010}}{a_3 - a_2} + \frac{w^{(2)}_{000} - w^{(2)}_{100}}{a_3 - a_2}$$

Then by (B.10)

$$\frac{w^{(1)}_{m-1,10} - w^{(1)}_{m10}}{a_2 - a_1} = \frac{w^{(2)}_{110} - w^{(2)}_{010}}{a_3 - a_2}. \tag{B.11}$$

Similarly, From (B.7)

$$\frac{w^{(1)}_{m-1,01} - w^{(1)}_{m01}}{a_2 - a_1} = \frac{w^{(2)}_{101} - w^{(2)}_{001}}{a_3 - a_2}. \tag{B.12}$$

Finally, from (B.9) we have

$$\frac{w^{(1)}_{m-1,11} - w^{(1)}_{m11}}{a_2 - a_1} = \frac{w^{(2)}_{111} - w^{(2)}_{011}}{a_3 - a_2} \tag{B.13}$$

(B.10)—(B.13) are the four x-direction collinear conditions. ◇

Theorem B.2 *If W_1 and W_2 in Lemma above are tri-cubic, and if both W_1 and W_2 interpolate C^3 data at the common four vertices of two adjacent cubes D_1 and D_2, then W_1 and W_2 are C^1 continuous on the common face of D_1 and D_2.*

Proof: From Lemma B.2, we know that at each common vertex, four x-direction collinear conditions are satisfied. Therefore, all the (sixteen) collinear conditions are satisfied. From (A.8) we know that W_1 and W_2 are C^1 continuous at the common face. ◇

Lemma B.3 *Let $R = [a_1, a_3] \times [b_1, b_3]$ be a rectangle in the plane (see Figure 5.1(a)), and $w_1, \ldots, w_9$ be values on the nine grid points. Then the six collinear conditions*

$$\frac{w_{i+1} - w_i}{a_2 - a_1} = \frac{w_{i+2} - w_{i+1}}{a_3 - a_2}, \quad i = 1, 4, 7 \tag{B.14}$$

$$\frac{w_{i+3} - w_i}{b_2 - b_1} = \frac{w_{i+6} - w_{i+3}}{b_3 - b_2}, \quad i = 1, 2, 3 \tag{B.15}$$

are satisfied if any five of them are satisfied.

Proof: Suppose (B.14) holds for $i = 4, 7$, (B.15) holds for $i = 1, 2, 3$. We show that (B.14) holds for $i = 1$. Since (see (A.9)) $w_5 = \frac{\Delta x_2 w_4}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1 w_6}{\Delta x_1 + \Delta x_2}$, $w_8 = \frac{\Delta x_2 w_7}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1 w_9}{\Delta x_1 + \Delta x_2}$, then

$$\begin{aligned} w_2 &= \frac{\Delta y_1}{\Delta y_2}(w_5 - w_8) + w_5 \\ &= \frac{\Delta y_1}{\Delta y_2}\left(\frac{\Delta x_2(w_4 - w_7)}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1(w_6 - w_9)}{\Delta x_1 + \Delta x_2}\right) + \left(\frac{\Delta x_2 w_4}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1 w_6}{\Delta x_1 + \Delta x_2}\right) \\ &= \frac{\Delta x_2}{\Delta x_1 + \Delta x_2}\left(\frac{\Delta y_1(w_4 - w_7)}{\Delta y_2} + w_4\right) + \frac{\Delta x_1}{\Delta x_1 + \Delta x_2}\left(\frac{\Delta y_1(w_6 - w_9)}{\Delta y_2} + w_6\right) \\ &= \frac{\Delta x_2 w_1}{\Delta x_1 + \Delta x_2} + \frac{\Delta x_1 w_1}{\Delta x_1 + \Delta x_2}. \end{aligned}$$

That is, (B.14) holds for $i = 1$. Similarly, any other equality can be deduced from the remaining five equalities. ◇

Lemma B.4 *Let $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$ be a given cube. At each of its eight vertices $P_{i_1 i_2 i_3}$, we are given C^3 data. If we subdivide D into eight sub-cubes $D_{i_1 i_2 i_3}$ (see Figure 5.1(b)), then there exists uniquely one piecewise function W on D such that*

a. *$W_{i_1 i_2 i_3} = W|_{D_{i_1 i_2 i_3}}$ is a polynomial of degree 2, that interpolates the set of C^3 data at $P_{i_1 i_2 i_3}$.*

b. *W is C^1 continuous on D.*

c. *If W' is defined in the same way over an adjacent cube D', then W and W' are C^1 continuous on the common face of D and D'.*

Proof: In the cube $D_{i_1 i_2 i_3}$, let

$$W_{i_1 i_2 i_3}(x, y, z) = \sum_{i=0}^{2} \sum_{j=0}^{2} \sum_{k=0}^{2} w_{ijk}^{i_1 i_2 i_3} B_i^2 B_j^2 B_k^2.$$

Then by Lemma 5.1 we know that eight coefficients around vertex $P_{i_1 i_2 i_3}$ are determined by interpolating the C^3 data. The remaining control points are determined by the collinear condition. For example, $w_{020}^{000} = \frac{\Delta y_2 w_{010}^{000}}{\Delta y_1 + \Delta y_2} + \frac{\Delta y_1 w_{010}^{010}}{\Delta y_1 + \Delta y_2}$. Therefore, W is C^1 continuous on D.

Now suppose $W'(x, y, z)$ is a function defined on D' in the same way as W, and D' and D share a common face. We shall prove that W and W' are C^1 continuous. Obviously, they have the same control points on the common face. Then by Lemma B.3, we know that collinear conditions are satisfied for each grid point on the common face. For example, consider a collinear condition at w_{020}^{000}. Suppose D' is x-neighbor of D then we follow Figure 5.1(b) where $(w_{110}^{000}, w_{120}^{000}, w_{110}^{010})$, $(w_{110}^{\prime 100}, w_{120}^{\prime 100}, w_{110}^{\prime 110})$, and $(w_{010}^{000}, w_{020}^{000}, w_{010}^{\prime 010})$ satisfy collinear conditions since they are on one cube. Furthermore $(w_{110}^{\prime 100}, w_{010}^{000}, w_{110}^{000})$ and $(w_{110}^{\prime 110}, w_{010}^{010}, w_{110}^{110})$ also satisfy the collinear conditions (see the proof of Lemma B.3). Then by Lemma B.3 the remaining collinear condition holds. ◇

Lemma B.5 *Let a, b, c, d and w be given (see Figure 6.1(a)) such that*

$$\frac{w - a}{\Delta x_1} = \frac{b - w}{\Delta x_2}, \quad \frac{w - c}{\Delta y_1} = \frac{d - w}{\Delta y_2} \tag{B.16}$$

Then there exist $w_1 \ldots w_4$ satisfying the equations

$$\frac{d - w_3}{\Delta x_1} = \frac{w_4 - d}{\Delta x_2}, \quad \frac{c - w_1}{\Delta x_1} = \frac{w_2 - c}{\Delta x_2}, \quad \frac{a - w_1}{\Delta y_1} = \frac{w_3 - a}{\Delta y_2}, \quad \frac{b - w_1}{\Delta y_1} = \frac{w_4 - b}{\Delta y_2} \tag{B.17}$$

and the solution can be expressed as

$$w_2 = \frac{\Delta x_1 + \Delta x_2}{\Delta x_1} c - \frac{\Delta x_2}{\Delta x_1} w_1 \tag{B.18}$$

$$w_3 = \frac{\Delta y_1 + \Delta y_2}{\Delta y_1} a - \frac{\Delta y_2}{\Delta y_1} w_1 \tag{B.19}$$

$$w_4 = \frac{\Delta x_1 + \Delta x_2}{\Delta x_1} d - \frac{\Delta x_2}{\Delta x_1} \frac{\Delta y_1 + \Delta y_2}{\Delta y_1} a + \frac{\Delta x_2 \Delta y_2}{\Delta x_1 \Delta y_1} w_1. \tag{B.20}$$

Proof: First, it is easy to see that the first three equations of (B.17) are equivalent to (B.18)—(B.20). Now we prove that the last equality of (B.17) holds if w_2—w_4 are defined by (B.18)—(B.20). It follows from (B.16) that $a = \frac{\Delta x_1+\Delta x_2}{\Delta x_2} w - \frac{\Delta x_1}{\Delta x_2} b$, $d = \frac{\Delta y_1+\Delta y_2}{\Delta y_1} w - \frac{\Delta y_2}{\Delta y_1} c$., Substituting these into (B.20), we have $w_4 = \frac{\Delta y_1+\Delta y_2}{\Delta y_1} b - \frac{\Delta y_2}{\Delta y_1} \frac{\Delta x_1+\Delta x_2}{\Delta x_1} c + \frac{\Delta y_2}{\Delta y_1} \frac{\Delta x_2}{\Delta x_1} w_1 = \frac{\Delta y_1+\Delta y_2}{\Delta y_1} b - \frac{\Delta y_2}{\Delta y_1} w_2$. Therefore the last equality of (B.17) holds. ◇

Lemma B.6 *For a given cube, let* $(q_1, z_1, q_2, y_2, q_3, z_2, q_4, y_1, w)$, $(r_1, z_1, r_2, x_2, r_3, z_2, r_4, x_1, w)$ *and* $(s_1, x_1, s_2, y_2, s_3, x_2, s_4, y_1, w)$ *satisfy the 2D-collinear conditions (see Figure 6.1(b)). Then there exist* $w_1, \ldots, w_8$ *satisfying twelve collinear conditions (each one corresponds to one edge of the cube), and the solution is one dimensional. More precisely, the set of twelve equations is equivalent to a subset of seven equations (E.g., four equations on a face and one from the opposite face can be removed).*

Proof: Let $[w_i w_j]$ denote the collinear equation for the unknowns w_i and w_j on the corresponding edge (see Figure 6.1(b)). Suppose we remove the equations $[w_2 w_3]$, $[w_3 w_7]$, $[w_6 w_7]$, $[w_2 w_6]$ on the top face of Figure 6.1(b) and $[w_5 w_8]$ on the bottom face. Then we have seven equations left and the eight unknowns $w_1, \ldots, w_8$. Now take w_1 to be free, we shall show that $w_2, \ldots, w_8$ can be determined from w_1 uniquely by collinear conditions.

First, it follows from Lemma B.5 that w_4, w_5, w_8 are uniquely defined by 2D-collinear conditions on the bottom face, and they are expressed explicitly by w_1.

Secondly, w_2 w_3 are similarly determined and expressed by w_1 by using the 2D-collinear condition on the face that face us in Figure 6.1(b). Similarly, w_3, w_7 are expressed explicitly by w_4, and w_4 is in turn expressed by w_1, then w_3, w_7 are expressed by w_1. In the same fashion, w_6 is expressed by w_1 also.

Finally, it is easy to show from Lemma B.5 that all the twelve collinear conditions are satisfied by such defined w's. Since each w is defined uniquely, then seven equations used are linear independent. They are therefore equivalent to the twelve equations. ◇