Interpolatory, solid subdivision of unstructured hexahedral meshes

Kevin T. McDonnell,
Yu-Sung Chang,
Hong Qin

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794–4400, USA
E-mail: {ktm, yusung, qin}@cs.sunysb.edu

This paper presents a new, volumetric subdivision scheme for interpolation of arbitrary hexahedral meshes. To date, nearly every existing volumetric subdivision scheme is approximating, i.e., with each application of the subdivision algorithm, the geometry shrinks away from its control mesh. Often, an approximating algorithm is undesirable and inappropriate, producing unsatisfactory results for certain applications in solid modeling and engineering design (e.g., finite element meshing). We address this lack of smooth, interpolatory subdivision algorithms by devising a new scheme founded upon the concept of tri-cubic Lagrange interpolating polynomials. We show that our algorithm is a natural generalization of the butterfly subdivision surface scheme to a tri-variate, volumetric setting.

Key words: Subdivision algorithms – Geometric and topological representations – Solid modeling – Multiresolution models – Volumetric meshes

1 Introduction and motivation

Since the pioneering work on procedural subdivision surface schemes by Catmull and Clark [3] and Doo and Sabin [6] in the late 1970s, much research has been undertaken to develop and analyze a myriad of other subdivision algorithms. The great majority of this work has involved surface schemes exclusively, primarily because of their many advantages, including:

– Generalization of tensor-product splines
– Unification of polygonal representation and curved geometry
– Multiresolution analysis and level-of-detail control
– Numerical stability and ease of implementation
– Representation of topologically complex geometric shapes
– Absence of complicated patching or trimming operations

In contrast, there has been substantially less exploration of subdivision solid algorithms, i.e., volumetric schemes. Such schemes are similar to surface algorithms in that they consist of a set of rules for refining control geometry as well as an algorithm for connecting the new vertices. Volumetric schemes, however, consist not only of rules for polygonal faces, edges and vertices, but also for polyhedral cells. It is the inclusion of the cells that leads such algorithms to be volumetric and hence, extremely useful in solid modeling, volumetric meshing, finite element analysis, and other relevant applications.

To our best knowledge, all existing subdivision solid schemes are approximating in nature with two exceptions. One is the interpolatory algorithm by Pascucci and Bajaj [16], which is a trivial tensor-product generalization of Dyn et al.'s four-point scheme [8] to rectilinear, volumetric grids. The other scheme is an algorithm we recently published [5] for recursive subdivision of meshes organized around octet-truss structures. With an approximating subdivision scheme, each application of the subdivision algorithm causes the geometry to *shrink* away from the initial control mesh. There are numerous applications in which an approximating subdivision solid scheme becomes unattractive and produces unsatisfactory results. Typical examples include finite element analysis (FEA) and computational fluid dynamics (CFD) simulation. In such applications, it is imperative that the material values (e.g., mass, velocity, pressure) not be modified as the domain space is refined, and that the simulation variables be interpolated smoothly across the elements. In addition,

one can easily find scenarios in CAD/CAM in which both geometric constraints and functional requirements must be satisfied exactly, and may be difficult or impossible to meet by using an approximating scheme. Another application is scattered data interpolation, in which the solution may be derived more easily by an interpolating algorithm.

1.1 Research contributions

To address the lack of general, interpolatory, volumetric subdivision algorithms, we propose, derive, analyze, and implement such an algorithm for recursively subdividing arbitrary hexahedral meshes. In contrast to tensor-product schemes, like that of Pascucci and Bajaj [16], our scheme can subdivide any hexahedral mesh, regardless of its topology and connectivity. Our algorithm seeks to bridge the gap between existing state-of-the-art subdivision algorithms and solid modeling applications in the real world. It is strongly inspired by existing interpolatory schemes for subdivision curves and surfaces and non-trivially generalizes such algorithms to a volumetric setting. The scheme, which is derived directly from tri-cubic, Lagrange, interpolating polynomials, can be used to subdivide hexahedral meshes to any user-specified level in order to conform with error tolerances or aesthetic requirements.

1.2 Background of subdivision solid schemes

Subdivision solids have recently emerged as a new solid modeling approach and interactive deformation technique. In comparison with well-established modeling techniques associated with subdivision surfaces, subdivision solid formulations transcend the conventional limitation of surface-based approaches by defining geometry and topology both in the interior and on the boundary of solid objects. The first documented volumetric subdivision algorithm, that of MacCracken and Joy [12], generalizes tri-cubic B-spline solids to meshes of arbitrary topology. Bajaj and colleagues [1] have proposed an alternative to the MacCracken–Joy algorithm that also reproduces tri-cubic B-spline volumes under regular topological conditions. Both algorithms are approximating in nature. Chang et al. [4] derived a C^1 continuous, volumetric subdivision solid scheme based on box splines that must be applied over hybrid

tetrahedral/octahedral meshes. Other recent work includes the investigation of wavelet decompositions using subdivision volumes [2], hierarchical representation of time-varying data [11], and physics-based animation and volumetric sculpting [13, 14].

2 From butterfly surfaces to interpolatory subdivision solids

Our subdivision solid algorithm is inspired by an extension of Dyn et al.'s butterfly subdivision scheme for interpolation of triangular meshes [9]. Specifically, the algorithm has its roots in the *averaged* butterfly scheme [7], which, unlike the normal butterfly algorithm, is applied over *quadrilateral* meshes. We have discovered that the original butterfly algorithm does not generalize directly to 3D, whereas the averaged butterfly scheme does. The reason can be explained as follows. In order to derive a volumetric subdivision scheme that contains a single subdivision rule each for vertices, edges, faces and cells, we need a mesh that has a single cell type. Unfortunately, a tetrahedral mesh fails to satisfy this requirement. Put another way, we cannot fill a 3D space using a single type of tetrahedron. However, in both 2D and 3D, the spaces can be completely filled in a topologically regular fashion using rectangular geometry (i.e., rectangles in 2D, hexahedra in 3D). The averaged butterfly scheme provides an approach for using such geometry to fill a 2D space in a recursive fashion. Our new algorithm provides the analogous case in 3D by recursively subdividing hexahedral meshes.

In the following sections we first derive the normal butterfly algorithm and then present a naive, asymmetric extension of that scheme to 3D. Afterwards, we present our subdivision scheme, which improves the naive refinement strategy by averaging multiple copies of the naive algorithm's masks. In this way, the derivation borrows the idea presented in [7], in which multiple copies of the normal butterfly algorithm's subdivision masks are averaged to derive the averaged butterfly scheme. This approach produces a symmetric, volumetric subdivision scheme for use over hexahedral meshes.

2.1 Butterfly subdivision surfaces

The derivation of the butterfly subdivision algorithm [7, 9] starts from the regular decomposition of

Fig. 1. Butterfly scheme mask for a regular 2D domain
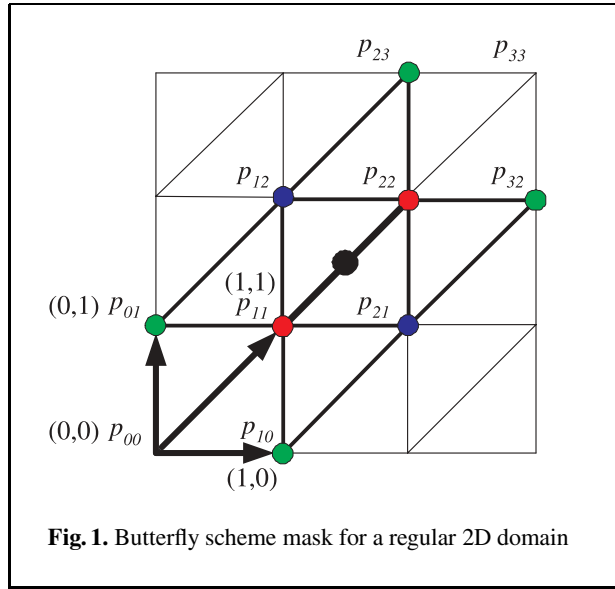
the 2D Euclidian space into a three-directional grid, as illustrated in Fig. 1. Without loss of generality we will assume the three grid directions are $(1, 0)$, $(0, 1)$ and $(1, 1)$. We restrict ourselves to the finite grid in the figure for sake of clarity. In the mathematical limit of subdivision, the algorithm reproduces bicubic surfaces that are at least C^1 continuous across surface patch boundaries.

Now, the cubic polynomial of two variables written in general form contains 10 terms:

$$f(x, y) = a_0 x^3 + a_1 y^3 + a_2 x^2 y + a_3 x y^2 + a_4 x^2$$
$$+ a_5 y^2 + a_6 x y + a_7 x + a_8 y + a_9. \quad (1)$$

Note that we will assume that f is a vector-valued function and returns a 2-vector. In order to derive an interpolating subdivision scheme that reproduces this polynomial over the regular grid shown in Fig. 1, it is necessary first to compute the coefficients of the function (i.e., the a_i's). Let $a = [a_0 \, a_1 \, \ldots \, a_9]^\top$ be the matrix of polynomial coefficients; let p be the column vector of the 10 grid positions labeled in Fig. 1; and let F be a square matrix where F_{ij} represents the ith term of $f(x, y)$ evaluated at the jth grid position. We will assume that the grid positions are enumerated in the following order:

$$p = [p_{00} \, p_{01} \, p_{10} \, p_{11} \, p_{12} \, p_{21} \, p_{22} \, p_{23} \, p_{32} \, p_{33}]^\top, \quad (2)$$

where an entry p_{ij} represents the vector

$$p_{ij} = [i \, j]^\top.$$

The system of equations $Fa = p$ cannot be solved exactly, however, because F is singular. It can be made non-singular by removing the two mixed terms from the polynomial, $x^2 y$ and $x y^2$, leaving eight terms. Let us call this new function $\widetilde{f}(x, y)$. Note that the extremal grid positions p_{00} and p_{33} are also removed. The modified system of equations $\widetilde{F}\widetilde{a} = \widetilde{p}$ can then be solved directly with Gaussian elimination. The coefficients we obtain ($\widetilde{a}$) are expressed in terms of the eight grid points.

Computing the position of a new vertex with this polynomial is a simple manner of evaluating $\widetilde{f}(x, y)$ at the center of the grid using the coefficients we just obtained. This evaluation produces the following subdivision rule:

$$\widetilde{f}\left(\frac{3}{2}, \frac{3}{2}\right) = \frac{1}{2}(p_{11} + p_{22}) + 2w(p_{12} + p_{21})$$
$$- w(p_{01} + p_{10} + p_{23} + p_{32}) \quad (3)$$

for the vertex labels used in Fig. 1 and for $w = \frac{1}{16}$. The weight, w, is the tension parameter of butterfly surfaces and can be modified to change the limit geometry. When applied over triangular surfaces, Eq. 3 is the familiar edge-bisection rule used in butterfly subdivision surface algorithm [9]. Since triangular meshes are generally preferred over quadrilateral meshes in computer graphics applications involving polygonal surfaces, the grid is typically tessellated. In the regular case, all main diagonals are oriented in the same direction. To obtain the edge-point masks for both horizontal and vertical edges, the mask in Fig. 1 can be mapped onto the other two directions, $(1, 0)$ and $(0, 1)$. For instance, the mask for a horizontal edge is depicted in Fig. 2. The mask for the $(0, 1)$ direction can be obtained in a similar fashion.

2.2 Naive extension of butterfly scheme to volumes

Based on the above derivation of the butterfly surface algorithm, it is straightforward, although certainly not trivial, to generalize the approach to a volumetric setting. The essential derivation process remains the same, although in the volumetric case, the underlying interpolating polynomial is tri-variate, rather
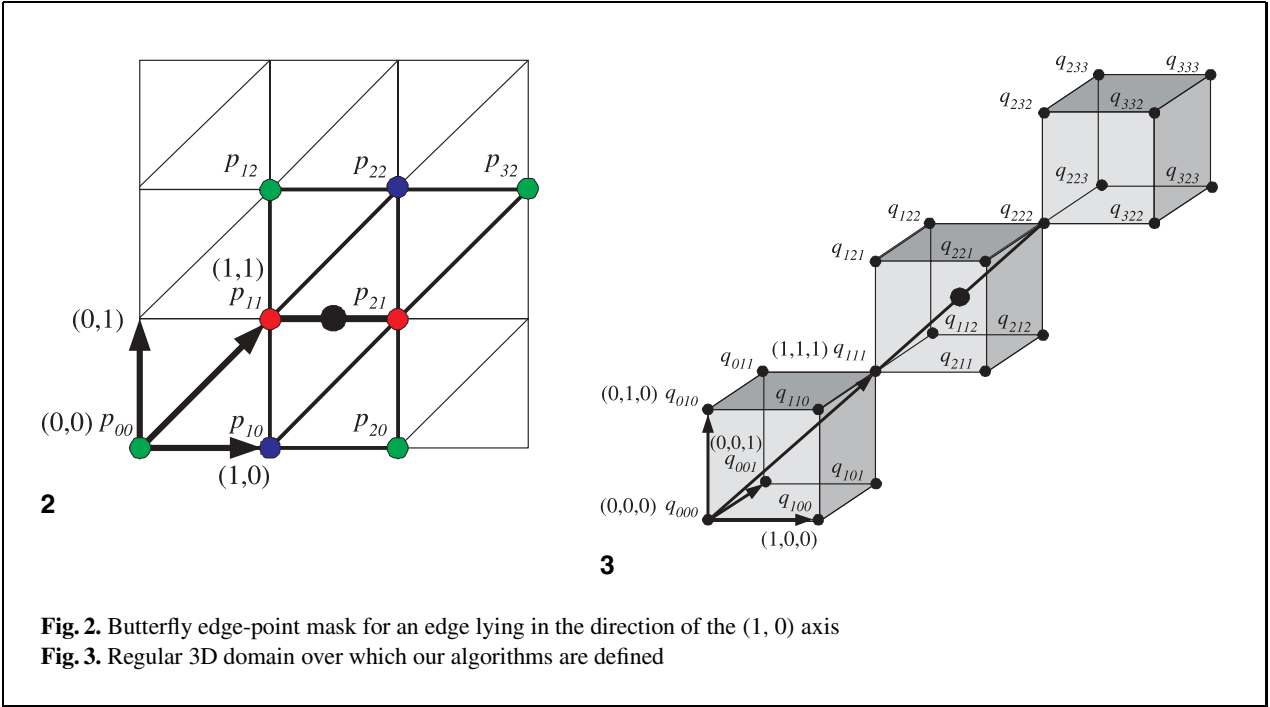
Fig. 2. Butterfly edge-point mask for an edge lying in the direction of the (1, 0) axis

Fig. 3. Regular 3D domain over which our algorithms are defined

than bi-variate. As a result, the same subdivision formula generates three subdivision rules, one each for edges, faces and cells. Additionally, the grid is comprised not of quadrilaterals, but of hexahedra. This has several ramifications for the subdivision rules (which will be explained later).

Let us begin the derivation by noting that the cubic polynomial of three variables written in general form contains 20 terms:
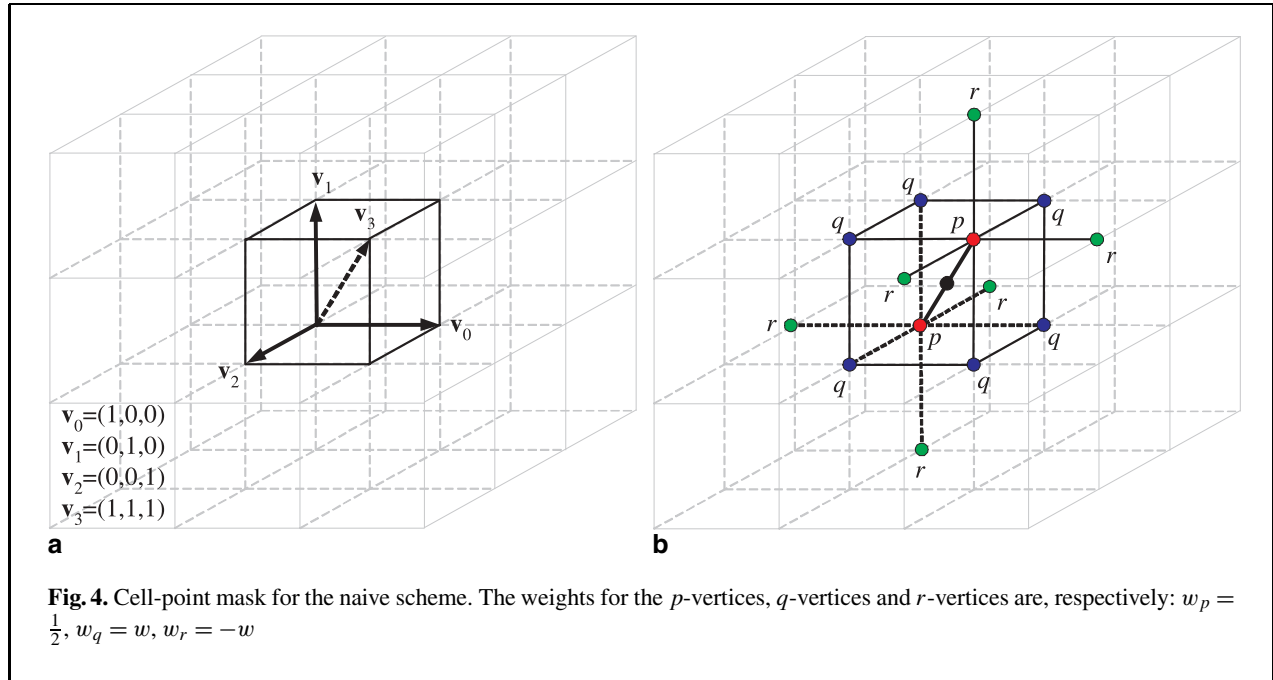
$$\begin{aligned} h(x, y, z) = &\, b_0 x^3 + b_1 y^3 + b_2 z^3 + b_3 x^2 y + b_4 x^2 z \\ &+ b_5 x y^2 + b_6 y^2 z + b_7 x z^2 + b_8 y z^2 + b_9 x y z \\ &+ b_{10} x^2 + b_{11} y^2 + b_{12} z^2 + b_{13} x y + b_{14} x z \\ &+ b_{15} y z + b_{16} x + b_{17} y + b_{18} z + b_{19}. \end{aligned} \quad (4)$$

Note that h is a vector-valued function and returns a 3-vector. (As we see later, however, $h(x, y, z)$ can be modified to interpolate material values and other scalars.) The corresponding regular grid for this polynomial can be seen in Fig. 3. By *regular* we mean that each vertex has valence six. As we saw in the butterfly surface derivation, all face diagonals must be oriented in the same direction for the grid to be considered regular. The same is true for the 3D domain, with the additional requirement that cell diagonals must also point in the same direction. Note

as well that the two extremal points, q_{000} and q_{333}, must not be considered in order to have exactly the 20 positions required for Eq. 4. As we did in the surface derivation, we assemble the polynomial coefficients into a column vector, b; we assemble the grid positions into a column vector, q; and then we compute a matrix H, where H_{ij} indicates the ith term of $h(x, y, z)$ evaluated at the jth grid position. We will assume that, as with the p vector earlier, the grid positions in q are enumerated in lexicographic order. Just as the matrix F was singular because of the mixed terms, so is H. This precludes a direct solution of $Hb = q$. After removing the six mixed terms of $h(x, y, z)$ (i.e., $x^2 y$, $x^2 z$, $x y^2$, $x z^2$, $y^2 z$, $y z^2$), we are left with a polynomial, $\tilde{h}(x, y, z)$, of 14 terms:

$$\begin{aligned} \tilde{h}(x, y, z) = &\, \tilde{b}_0 x^3 + \tilde{b}_1 y^3 + \tilde{b}_2 z^3 + \tilde{b}_3 x y z \\ &+ \tilde{b}_4 x^2 + \tilde{b}_5 y^2 + \tilde{b}_6 z^2 + \tilde{b}_7 x y + \tilde{b}_8 x z \\ &+ \tilde{b}_9 y z + \tilde{b}_{10} x + \tilde{b}_{11} y + \tilde{b}_{12} z + \tilde{b}_{13}. \end{aligned} \quad (5)$$

Note that we also drop the six grid positions q_{001}, q_{010}, q_{100}, q_{233}, q_{323} and q_{332}. The smaller set of equations $\tilde{H}\tilde{b} = \tilde{q}$ can then be solved directly to obtain the 14 coefficients in $\tilde{b}$. When $\tilde{h}(x, y, z)$ is evaluated at the center of the grid with these co-

Fig. 4. Cell-point mask for the naive scheme. The weights for the p-vertices, q-vertices and r-vertices are, respectively: $w_p = \frac{1}{2}, w_q = w, w_r = -w$

efficients, we obtain the following subdivision formula:

$$\widetilde{h}\left(\frac{3}{2}, \frac{3}{2}, \frac{3}{2}\right) = \frac{1}{2} \sum_{i=1}^{2} p_i + w \sum_{i=1}^{6} q_i - w \sum_{i=1}^{6} r_i \quad (6)$$
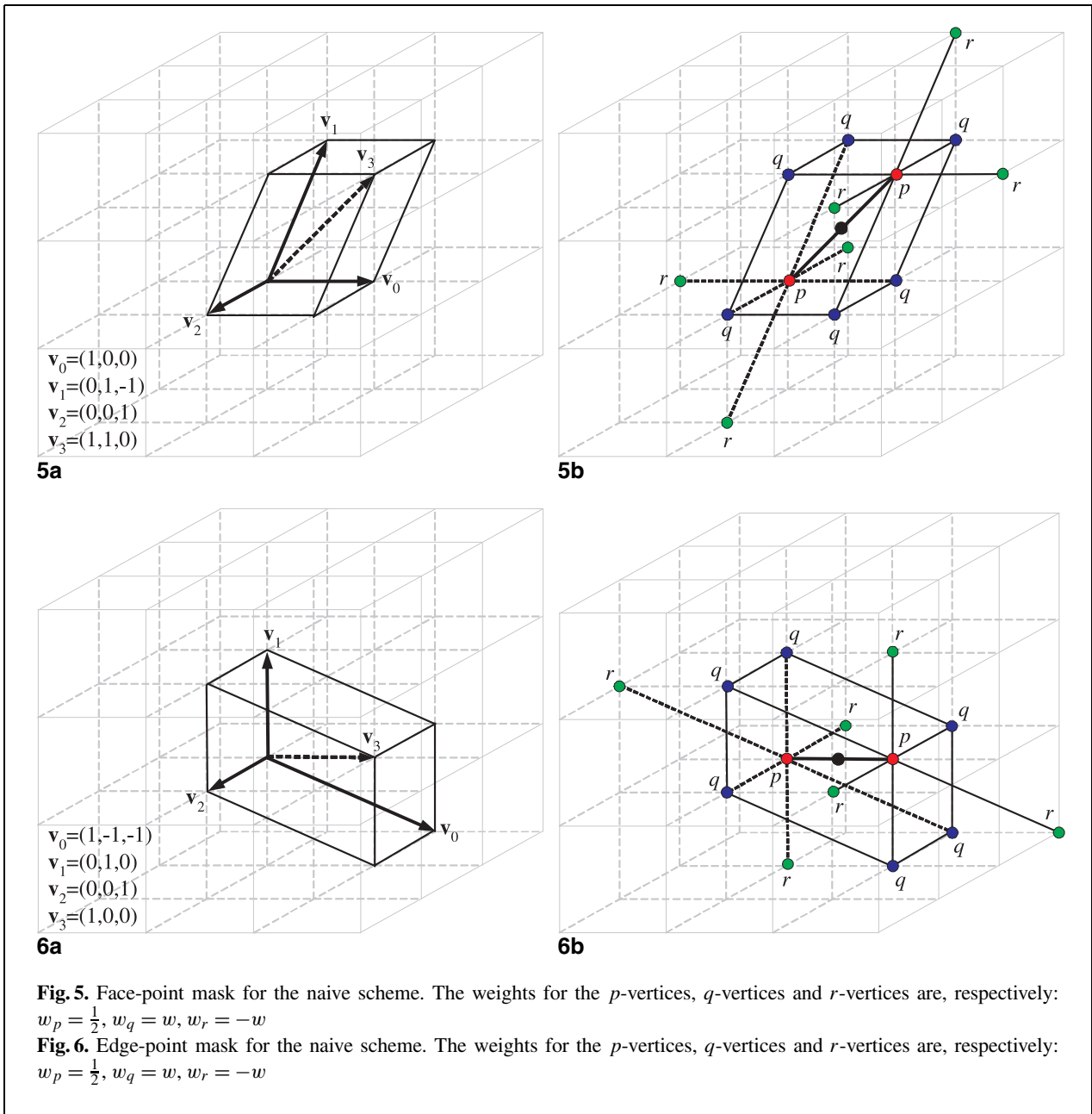
for the vertex labels used in Fig. 4b and for $w = \frac{1}{16}$.

Equation 6 and the corresponding grid configuration of Fig. 4 define the subdivision formula for a naive generalization of the original butterfly surface scheme to 3D. In particular, Eq. 6 defines the rule to compute a cell-point whose mask is exactly the vertex configuration depicted in Fig. 4b. (Henceforth, all subdivision masks are drawn in a right-handed coordinate system.) It is interesting to note that the butterfly algorithm generalizes in 3D to a cubical grid and not a tetrahedral one, for the reasons given in Sect. 2.

In the butterfly surface algorithm, the original 8-point mask for the (1, 1) direction is mapped onto the (1, 0) and (0, 1) directions to compute the other subdivision masks. In our 3D algorithm, the cell-point mask can be mapped in a similar fashion to define the face-point and edge-point masks. Specifically, the face-point rules can be acquired by mapping the cell-point rule onto the directions (0, 1, 1),

(1, 0, 1) and (1, 1, 0). (There are six face-point rules in total, but the remaining three can be acquired via symmetry from the three enumerated cases.) This mapping takes place as follows: note that for the cell-point rule, the major directions that define the cell are (1, 0, 0), (0, 1, 0) and (0, 0, 1), and its main diagonal is the sum of these directions: (1, 1, 1) (see Fig. 4a). In order to define the face-point rule for the front-facing and back-facing faces in Fig. 4, we map the three directions as follows: (1, 0, 0) → (1, 0, 0), (0, 1, 0) → (0, 1, −1) and (0, 0, 1) → (0, 0, 1). Figure 5a shows this mapping clearly. When we add the three new directions, we obtain the vector (1, 1, 0), which corresponds with the bold line in Fig. 5b. A similar process can be followed to map the cell-point mask to obtain other face-point masks.

The edge-point rules can be acquired by mapping the cell-point rule onto the directions (0, 0, 1), (0, 1, 0) and (1, 0, 0). There is a total of 12 edge-point rules, but the other nine edge-point masks can be acquired via symmetry from these three cases. For the particular edge-point mask given in Fig. 6b, the mapping is as follows: (1, 0, 0) → (1, −1, −1), (0, 1, 0) → (0, 1, 0) and (0, 0, 1) → (0, 0, 1). When we add the new directions, we obtain the vector (1, 0, 0), which corresponds to the bold line

Fig. 5. Face-point mask for the naive scheme. The weights for the p-vertices, q-vertices and r-vertices are, respectively: $w_p = \frac{1}{2}, w_q = w, w_r = -w$

Fig. 6. Edge-point mask for the naive scheme. The weights for the p-vertices, q-vertices and r-vertices are, respectively: $w_p = \frac{1}{2}, w_q = w, w_r = -w$

in Fig. 6b. The edge-point masks for the $(0, 1, 0)$ and $(0, 0, 1)$ directions can be found in a similar fashion.

The subdivision rule that defines the positions of the face-points and edge-points is the same as that of the cell-points, namely, Eq. 6. Note that the vertices have been labeled in Figs. 5b and 6b to correspond with the subdivision formula.

3 The subdivision algorithm

The naive 3D extension of the butterfly algorithm described in the previous section suffers from a number of drawbacks that hampers its usefulness. First, the main cell diagonals must be oriented in the same direction, and each vertex must have valence six. A strict vertex ordering is therefore required to main-

Fig. 7. Cylindrical control mesh subdivided with our subdivision solid scheme. **a** Control mesh. **b** Scaling the control mesh reveals the cells comprising the model. **c,d** Scaled subdivision mesh after one and two levels of subdivision, respectively
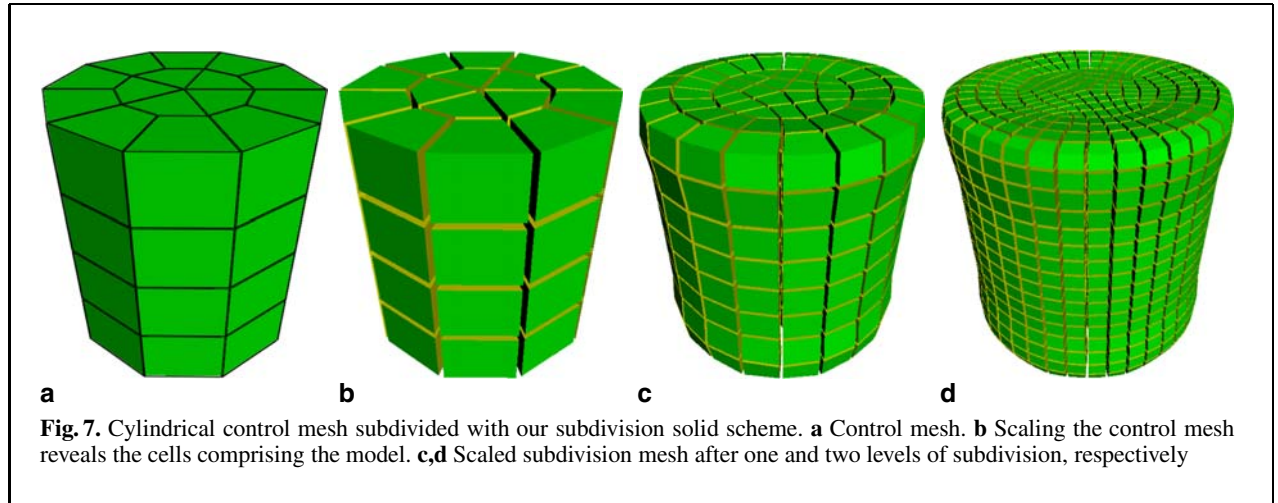
tain the diagonals between subdivision levels, which requires extra bookkeeping effort in the implementation. Second, and more importantly, the subdivision masks cannot accommodate arbitrary hexahedral meshes. This precludes their use in most real-world applications involving hexahedral meshes (e.g., FEA and CFD). Third, the subdivision masks are asymmetric and do not assign equal weight to all vertices in a local neighborhood. In particular, the masks are biased toward vertices lying at the ends of the cell diagonals. This causes certain vertices to influence the limit shape more than others, which is clearly undesirable in most situations.

In order to overcome these difficulties, we have derived an alternate scheme that averages multiple copies of the naive algorithm's subdivision masks. The approach we take is to compute the subdivision masks for all possible orientations of the main cell diagonal, add the contributions from each mask, and then normalize the results. This process is carried out for the cell-point mask, face-point mask and edge-point mask. As we show, the resulting subdivision masks, although larger in support, are symmetric and are more amenable to application over arbitrary hexahedral grids. A similar approach was taken by Dyn et al. [7] in deriving an averaged butterfly scheme for subdividing arbitrary quadrilateral meshes. An example of a mesh subdivided with our averaged scheme can be seen in Fig. 7.

3.1 Rules for meshes of regular topology

First we present subdivision rules for regular hexahedral meshes, in which each vertex in the mesh has va-

lence six. In Sect. 3.2 we generalize the rules to handle topologically non-regular hexahedral meshes.

3.1.1 Cell-point rule

The cell-point subdivision mask for our scheme is obtained by computing the different naive scheme's cell-masks (Fig. 4b) for the four possible orientations of the main diagonal in a cell. Adding these masks and normalizing the weights produces the averaged mask shown in Fig. 8. The cell-point rule is:

$$c_p = \frac{6w+1}{8} \sum_{i=1}^{8} p_i - \frac{w}{4} \sum_{i=1}^{24} q_i. \tag{7}$$

3.1.2 Face-point rule

Averaging the six naive scheme's face-point masks (Fig. 5b) produces the averaged mask seen in Fig. 9. The face-point rule is:

$$f_p = \frac{2w+1}{4} \sum_{i=1}^{4} p_i + \frac{w}{4} \sum_{i=1}^{8} q_i$$
$$- \frac{w}{4} \sum_{i=1}^{8} r_i - \frac{w}{8} \sum_{i=1}^{16} s_i. \tag{8}$$

3.1.3 Edge-point rule

Averaging the 12 naive scheme's edge-point masks (Fig. 6b) produces the averaged mask seen in Fig. 10. Note that many of the terms cancel each other, which results in a mask with fewer vertices than one might
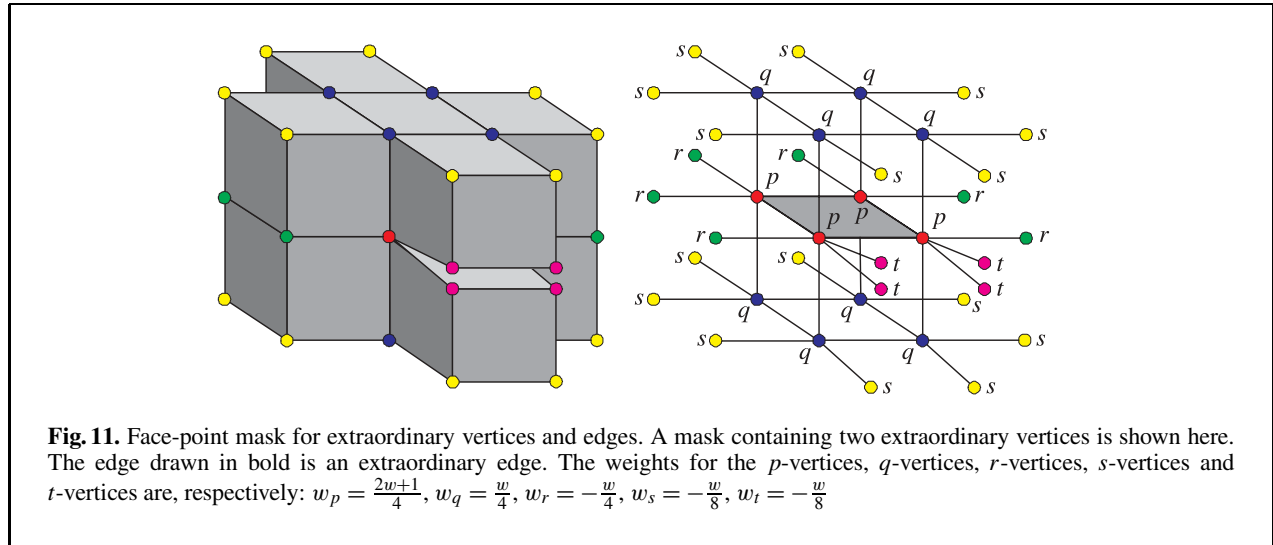
Fig. 8. Cell-point mask for our new subdivision scheme. The weights for the p-vertices and q-vertices are, respectively: $w_p = \frac{6w+1}{8}$, $w_q = -\frac{w}{4}$. The opaque rendering on the left shows the cellular structure of the mask

Fig. 9. Face-point mask for our new subdivision scheme. The weights for the p-vertices, q-vertices, r-vertices and s-vertices are, respectively: $w_p = \frac{2w+1}{4}$, $w_q = \frac{w}{4}$, $w_r = -\frac{w}{4}$, $w_s = -\frac{w}{8}$

Fig. 10. Edge-point mask for our new subdivision scheme. The weights for the p-vertices, q-vertices and r-vertices are, respectively: $w_p = \frac{1}{2}$, $w_q = \frac{w}{4}$, $w_r = -\frac{w}{4}$. The cancelling of terms in the derivation leads to a mask of smaller size than expected

expect. The edge-point rule is:

$$e_p = \frac{1}{2}\sum_{i=1}^{2} p_i + \frac{w}{4}\sum_{i=1}^{8} q_i - \frac{w}{4}\sum_{i=1}^{8} r_i. \qquad (9)$$

3.2 Rules for meshes of non-regular topology

As we show below, our new subdivision scheme is very amenable to application over arbitrary hexahe-dral meshes since it does not depend on a particular ordering of the vertices or choice of cell diagonal. The rules must be generalized to handle hexahe-dral meshes that feature *extraordinary edges* (i.e., edges with greater than four or less than four ad-jacent faces) and *extraordinary vertices* (i.e., ver-tices not of valence six). Fortunately, the rules can be generalized easily to handle these non-regular topological conditions. We now investigate how the rules must be changed to accommodate such situations.

Fig. 11. Face-point mask for extraordinary vertices and edges. A mask containing two extraordinary vertices is shown here. The edge drawn in bold is an extraordinary edge. The weights for the p-vertices, q-vertices, r-vertices, s-vertices and t-vertices are, respectively: $w_p = \frac{2w+1}{4}$, $w_q = \frac{w}{4}$, $w_r = -\frac{w}{4}$, $w_s = -\frac{w}{8}$, $w_t = -\frac{w}{8}$

3.2.4 Cell-point rule

In order to define subdivision rules and masks that can be applied over non-regular hexahedral meshes, we must examine the subdivision masks and determine which vertices, if any, are *shared* by cells that define the mask. We must also determine if the duplication or absence of shared vertices in a non-regular mesh causes the subdivision masks to be ill-defined. For instance, inspection of the cell-point mask in Fig. 8 reveals that the vertices adjacent to the cell itself can be identified *solely* by locating those vertices 1-adjacent to the cell's vertices. That is, the subdivision mask is obtained by taking vertices from the cell itself, as well as the cells *face-adjacent* to the cell. This means that Eq. 7 can be used to compute a cell-point even when the neighboring connectivity is very complicated. Hence, the cell-point subdivision mask does not require modification in order to handle arbitrary hexahedral meshes.

3.2.5 Face-point rule

Unlike the cell-point rule, the averaged scheme's face-point rule does require modification in order to handle extraordinary edges. As illustrated in Fig. 9, there are several vertices labeled r in the middle of the mask that are shared by adjacent cells that define the mask. Under a regular topological setting, there are eight such r-vertices, two per edge. If one or more of the edges connecting these vertices is extraordinary, which is the case in Fig. 11, then a special rule is required to handle the extra vertices introduced

into (or subtracted from) the mask. Note that vertices labeled q, although shared by adjoining cells, can be uniquely identified by obtaining them from the two cells that meet at the face itself. Since we assume that each face has at most two adjacent cells, there is never a problem in locating these eight q-vertices.

As we mentioned, there are eight r-vertices in the regular case of the face-point mask. Irregularities arise when one or more of the vertices p is extraordinary. Specifically, for each edge $\overline{p_i p_j}$ that is adjacent to more than four cells, the weight of each such r-vertex (now indicated by t in Fig. 11) becomes $-\frac{w}{8}$. Note that there will be at most two such vertices per end-point per edge since each face is shared by at most two cells (Fig. 11). The subdivision formula for the face-point is therefore modified as follows:

$$\widehat{f}_p = \frac{2w+1}{4} \sum_{i=1}^{4} p_i + \frac{w}{4} \sum_{i=1}^{8} q_i - \frac{w}{4} \sum_{i=1}^{8-2N} r_i$$

$$- \frac{w}{8} \sum_{i=1}^{16} s_i - \frac{w}{8} \sum_{i=1}^{4N} t_i \qquad (10)$$

where N indicates the number of extraordinary edges in the face. It is easy to confirm that this rule reduces to Eq. 8 when the mesh is regular.

3.2.6 Edge-point rule

The averaged scheme's edge-point mask must also be modified in order to handle two possible irregularities: (1) More than (or less than) four faces are
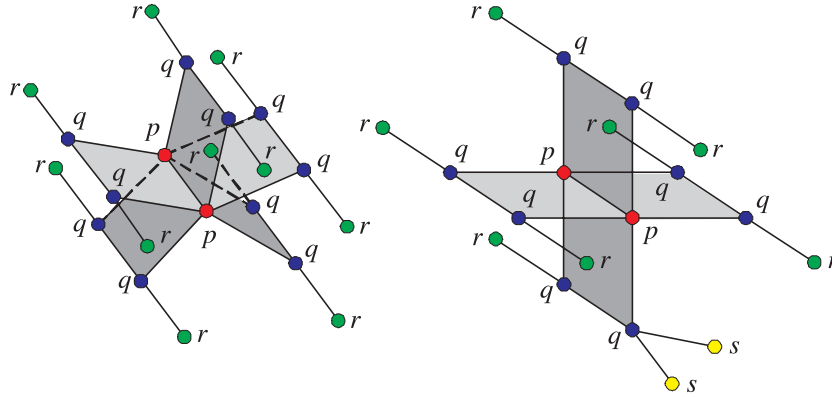
Fig. 12. Edge-point masks for non-regular topological settings. **a** Extraordinary edge. The weights for the p-vertices, q-vertices and r-vertices are, respectively: $w_p = \frac{1}{2}$, $w_q = \frac{w}{5}$, $w_r = -\frac{w}{5}$. **b** Regular edge containing one extraordinary vertex. The weights for the p-vertices, q-vertices, r-vertices and s-vertices are, respectively: $w_p = \frac{1}{2}$, $w_q = \frac{w}{4}$, $w_r = -\frac{w}{4}$, $w_s = -\frac{w}{8}$.
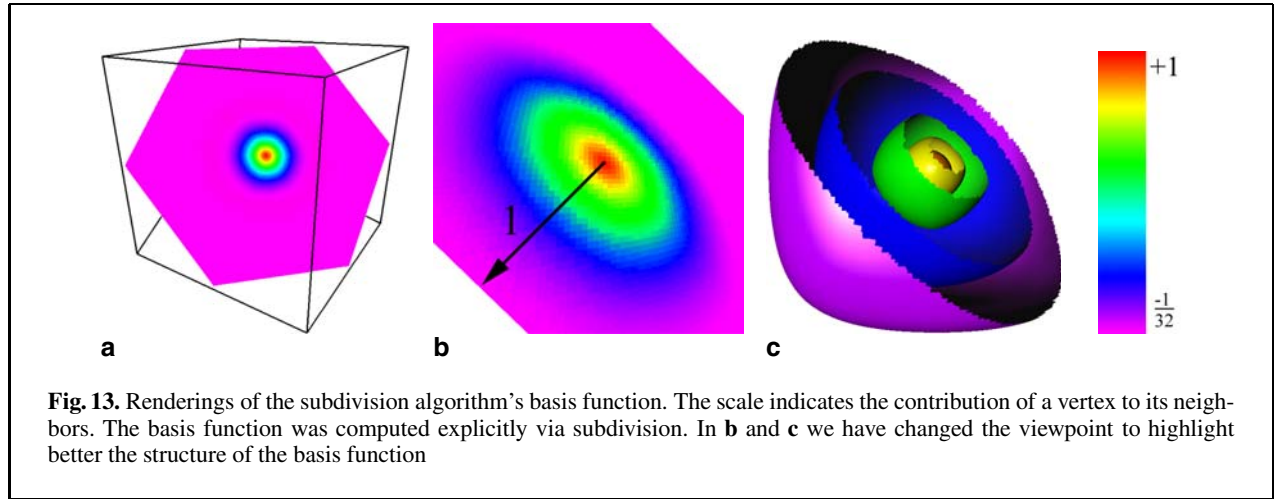
incident on the edge (Fig. 12a) and/or (2) One or more adjacent edge is extraordinary (Fig. 12b). The first situation arises when one or both of the edge's end-points are extraordinary vertices. The second circumstance appears when a vertex normally shared by adjacent cells (i.e., an r-vertex in Fig. 10) is replaced by two vertices because of a local topological irregularity. We call this kind of vertex a *split* vertex. (Recall that a similar situation can occur with the face-point mask when an extraordinary edge causes a normally shared vertex to be replaced by two distinct vertices; see Fig. 11.) In the regular case of four incident faces, each p-vertex receives a weight of $\frac{w}{4}$ and each r-vertex receives $-\frac{w}{4}$ (Fig. 10). For the general case of N incident faces, each p_i is given weight $\frac{w}{N}$, while each r_i receives $-\frac{w}{N}$. Figure 12a illustrates an edge with five incident faces and the resulting vertex weights. In Fig. 12b we see that the edge drawn in bold in the middle of the mask has a non-regular number of adjacent edges, which causes the weights of split vertices (labeled s) to change from $-\frac{w}{4}$ to $-\frac{w}{8}$. In the general case of N incident faces, this weight for a split vertex is $-\frac{w}{2N}$. These two non-regular topological conditions are subsumed by the following modified edge-point formula:

$$\widehat{e}_p = \frac{1}{2} \sum_{i=1}^{2} p_i + \frac{w}{N} \sum_{i=1}^{2N} q_i - \frac{w}{N} \sum_{i=1}^{2N-M} r_i - \frac{w}{2N} \sum_{i=1}^{2M} s_i$$

(11)

where N is the number of faces incident on the edge and M is the number of extraordinary edges incident on the p_i's that appear in the mask (i.e., the number of *split* vertices). Note that Eq. 11 reduces to Eq. 9 for regular meshes (i.e., $N = 4$, $M = 0$).

4 Boundary rules

For the examples in this paper, we employ Dyn's averaged butterfly scheme for quadrilateral meshes [7] to subdivide edges and faces on the boundaries of models. Our subdivision rules themselves do not make any assumption about the boundary geometry. This permits users of our algorithms to employ nearly any quadrilateral-based subdivision surface on the boundary to meet design requirements or aesthetic criteria. However, we suggest an interpolating scheme like the averaged butterfly algorithm be used in order to avoid self-intersection. Cells on the boundary require special care since their subdivision masks are ill-defined. Since there are not enough neighboring cells to specify the cell-point, face-point and edge-points masks completely, special rules must be applied in these regions. In our examples we simply employ linear subdivision to handle these cases. With each level of subdivision, the thickness of this special layer of cells decreases by one-half. This ensures that any error introduced by linear subdivision decreases exponentially as a func-

Fig. 13. Renderings of the subdivision algorithm's basis function. The scale indicates the contribution of a vertex to its neighbors. The basis function was computed explicitly via subdivision. In **b** and **c** we have changed the viewpoint to highlight better the structure of the basis function

tion of subdivision level. Note that the use of special rules on or near the boundary necessarily changes the continuity of the scheme in these small regions. This effect is unavoidable.

5 Proof of continuity

5.1 Convergence and continuity for meshes of regular topology

We will use techniques described by Dyn et al. [10] to prove that our subdivision scheme is C^1 continuous over hexahedral meshes of regular topology. The surface representation of our scheme is based on the averaged butterfly scheme, whose continuity was proven to be of the C^1 class by Dyn et al. [7] Therefore, we focus on the proof of the continuity of the solid case in this section.

Since our subdivision scheme has no closed-form expression for its basis functions (Fig. 13), we cannot simply extract the basis functions and examine them analytically. Therefore, we rely on the analysis of subdivision matrices and characteristic functions to study the scheme's convergence and continuity properties. By showing that the characteristic polynomials of the subdivision process have certain properties, we will demonstrate that the algorithm generates volumes that are C^1 in the limit.

Note that a subdivision algorithm can be expressed in matrix form as $p^{k+1} = S p^k$, where p^k is the vector of points at subdivision level k, S is the local subdivision mask, and p^{k+1} is the resulting vector of new

points. We further consider the subdivision process as discrete convolution of sequences (see [18]). In this way, we can relate the subdivision process with a polynomial expression, i.e., its *generating function*. Generally, any binary stationary subdivision scheme for solids can be written as

$$P_{k+1}(z) = a(z) P_k(z^2), \qquad z \in \mathbb{R}^3, \qquad (12)$$

where $P_k(z) = \sum_{\mu \in \mathbb{Z}^3} p_\mu^k z^\mu$ is a formal generating function associated with the control points $p^k = \{p_\mu^k\}_{\mu \in \mathbb{Z}^3}$ at the level k, and $a(z)$ is the characteristic polynomial derived from the local subdivision matrix S:

$$a(z) = \sum_{\mu \in \mathbb{Z}^3} a_\mu z^\mu. \qquad (13)$$

We follow a standard multi-index notation throughout the proof.

By comparing these coefficients after n iterations of the subdivision process, one can show (see [7]) that

$$\|S^n\|_\infty = \max_\gamma \sum_{\nu \in \mathbb{Z}^3} a_{2^n \nu + \gamma}^{[n]}, \qquad (14)$$

where $\gamma \in \{0, 1, \dots, 2^n - 1\}^3$ and $a^{[n]}(z) = \prod_{j=0}^{n-1} a(z^{2^j}) = \sum_\mu a_\mu^{[n]} z^\mu$. This relation will be used to calculate the norm of the subdivision matrix. We utilize the coefficients of the characteristic *polynomial* (or *Laurent polynomial*, to be correct) instead of the matrix itself to compute the norm. The rest of the proof will follow the steps below:

1. Find the characteristic polynomial of the scheme.
2. Derive the *difference processes* of the scheme along the directions that are associated with the characteristic polynomial of the scheme.
3. Prove that the difference processes are continuous by using their characteristic polynomials, and thereby show that the scheme is C^1.

Steps 2 and 3 are special cases of the following two theorems. Readers who are interested in the proofs of the theorems are referred to the work by Dyn et al. [10]

Theorem 1. *Let the characteristic polynomial of* S *have the form*

$$a(z) = q(z) \prod_{i=1}^{s} \left(z^{\theta^{(i)}} + 1\right), \tag{15}$$

where q is a Laurent polynomial and $\theta^{(i)} \in \mathbb{Z}^s$ *satisfies*

$$\left|det\left(\theta^{(1)}, \ldots, \theta^{(s)}\right)\right| = 1. \tag{16}$$

Let $\boldsymbol{D}_i$ *be the subdivision matrix corresponding to the polynomial* $a(z)(z^{\theta^{(i)}} + 1)^{-1}$. *Then the subdivision scheme associated with* S *is uniformly convergent if and only if for some* $L \in \mathbb{Z}_+$,

$$\|\boldsymbol{D}^L\|_\infty = \max_{1 \le i \le s} \|\boldsymbol{D}_i^L\|_\infty < 1. \tag{17}$$

Theorem 2. *Let* S *be convergent with a characteristic polynomial*

$$a(z) = (z^\theta + 1)^\nu 2^{-\nu} q(z), \tag{18}$$

where $\theta \in \mathbb{Z}^s$, $\nu \in \mathbb{Z}_+$, *and q is a Laurent polynomial. If the subdivision scheme associated with q converges uniformly, then for all initial control points* $\boldsymbol{p}^0$,

$$\partial_\theta^\nu S^\infty \boldsymbol{p}^0 \in C(\mathbb{R}^s), \tag{19}$$

where ∂_θ *means the directional derivative in the direction* θ, *i.e.,*

$$\partial_\theta f(x) = \lim_{t \to 0} \left(f(x + t\theta) - f(x)\right). \tag{20}$$

Theorems 1 and 2 provide us the sufficient conditions to guarantee the C^1 continuity of the subdivision scheme. The conditions for the norm of the matrix will be confirmed by means of the relation explained in Eq. 14. During most of the process, we will rely on numerical experiments to verify the satisfaction of the conditions.

The characteristic polynomial of the subdivision scheme can be computed by successive applications of the schemes over a regular mesh in 3D. It has the form of

$$a(z) = \sum_{\mu \in \mathbb{Z}^3} a_\mu z^\mu, \tag{21}$$

where the coefficients are given by

$$a_\mu = \frac{6w + 1}{8}, \qquad \mu = (\pm 1, \pm 1, \pm 1)$$

$$a_\mu = -\frac{w}{4}, \qquad \mu = (\pm 1, \pm 1, \pm 3)$$

$$a_\mu = \frac{2w + 1}{4}, \qquad \mu = (0, \pm 1, \pm 1)$$

$$a_\mu = \frac{1}{4}, \qquad \mu = (\pm 1, \pm 1, \pm 2)$$

$$a_\mu = -\frac{w}{4}, \qquad \mu = (0, \pm 1, \pm 3)$$

$$a_\mu = -\frac{w}{8}, \qquad \mu = (\pm 1, \pm 2, \pm 3)$$

$$a_\mu = \frac{1}{2}, \qquad \mu = (0, 0, \pm 1)$$

$$a_\mu = \frac{w}{4}, \qquad \mu = (0, \pm 1, \pm 2)$$

$$a_\mu = -\frac{w}{4}, \qquad \mu = (0, \pm 2, \pm 3)$$

$$a_{\mu'} = a_\mu, \qquad \text{if} \quad \mu' = \sigma(\mu), \quad \sigma \in S_3.$$

Here, S_3 denotes the set of all permutations over $\{1, 2, 3\}$, which is followed by the symmetry of the subdivision mask.

It is relatively easy to confirm that the scheme is convergent by means of eigenvalue analysis of the subdivision matrix. In particular, the subdominant eigenvalue of our subdivision scheme is strictly less than 1 for $w < 0.5$, which is sufficient to show its convergence. The characteristic polynomial of the subdivision scheme can be factored by $(1 + z_1)^2(1 + z_2)^2(1 + z_3)^2$. Therefore, it can be written as

$$a(z) = \frac{1}{2}(1 + z_1)^2(1 + z_2)^2(1 + z_3)^2 q(z, w), \tag{22}$$

where $q(z, w)$ is a Laurent polynomial with respect to z for a given weight w.

Now, we employ Theorems 1 and 2 for the proof of C^1 continuity. A close inspection of the theorems reveals that, for the scheme to be C^1 continuous, it is sufficient to show that $\|\boldsymbol{D}_{(i_1,i_2)}^L\|_\infty < 1$ for some L where $\boldsymbol{D}_{(i_1,i_2)}$ is a subdivision matrix whose generating function is

$$d_{(i_1,i_2)}(z) = 2(1+z_{i_1})^{-1}(1+z_{i_2})^{-1}\, a(z).$$

Because $a(z)$ is invariant over a permutation on indices, it is equivalent to show that $\|\boldsymbol{D}_{(1,1)}^L\| < 1$ and $\|\boldsymbol{D}_{(1,2)}^L\| < 1$, where

$$d_{(1,1)}(z) = 2(1+z_1)^{-2}\, a(z), \text{ and}$$
$$d_{(1,2)}(z) = 2(1+z_1)^{-1}(1+z_2)^{-1}\, a(z),$$

respectively.
If we let $\|\boldsymbol{D}^k\|_\infty = \max_{(i_1,i_2)} \|\boldsymbol{D}_{(i_1,i_2)}^k\|_\infty$, then when $w = \frac{1}{16}$, the norms are

$$\|\boldsymbol{D}^1\|_\infty = 1.75,$$

$$\|\boldsymbol{D}^2\|_\infty \simeq 1.5313,$$

$$\|\boldsymbol{D}^3\|_\infty \simeq 1.3523,$$

$$\|\boldsymbol{D}^4\|_\infty \simeq 1.0914,$$

$$\|\boldsymbol{D}^5\|_\infty \simeq 0.8188.$$

Figure 14 shows $\|\boldsymbol{D}^k\|_\infty$ as a function of the weight w. Generally, $\|\boldsymbol{D}^L\|_\infty < 1$ when $L \geq 5$, at least for $w \in (0, 0.1787]$. By the theorems, we can guarantee that, when w is within the range, the subdivision scheme is C^1 continuous over regular hexahedral meshes.

5.2 Continuity for meshes of non-regular topology

Non-regular topologies include cases in which a vertex or an edge has non-standard connectivity. In regular hexahedral meshes, each vertex has a valence of six and each edge is shared by four adjacent faces. When the mesh does not have these properties, we say that mesh has an *extraordinary vertex* or *extraordinary edge*. Because non-regular topologies become isolated from each other during the subdivision process, we can assume that we have only a finite number of extraordinary cases in any given mesh.
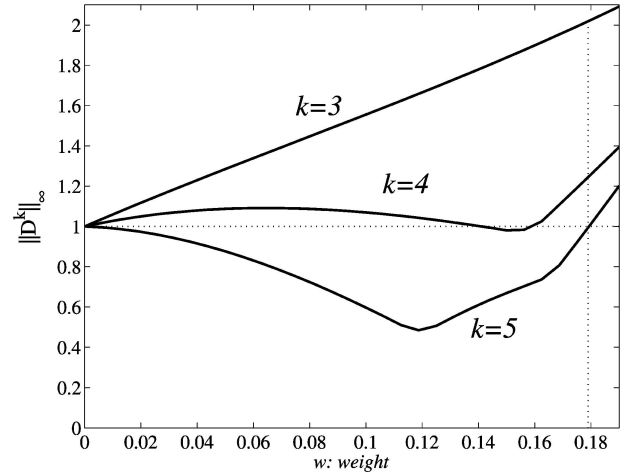For subdivision surface schemes, eigenanalysis is the standard technique to prove the continuity of the

Fig. 14. A graph of $\|\boldsymbol{D}^k\|_\infty$ with respect to the weight value w

scheme around non-regular topologies. Suppose λ_i's are the eigenvalues of the subdivision matrix $\boldsymbol{S}$ in decreasing order. The basic idea is that the initial control points $\boldsymbol{p}^0$ can be expressed by the corresponding eigenvectors $\boldsymbol{v}_i$ in the eigenspace of the matrix $\boldsymbol{S}$,

$$\boldsymbol{p}^0 = \boldsymbol{a}_0\boldsymbol{v}_0 + \boldsymbol{a}_1\boldsymbol{v}_1 + \cdots + \boldsymbol{a}_n\boldsymbol{v}_n, \qquad (23)$$

and the limit process can be expressed as

$$\boldsymbol{p}^\ell = \boldsymbol{S}^\ell\boldsymbol{p}^0 = \lambda_0^\ell\boldsymbol{a}_0\boldsymbol{v}_0 + \lambda_1^\ell\boldsymbol{a}_1\boldsymbol{v}_1 + \cdots + \lambda_n^\ell\boldsymbol{a}_n\boldsymbol{v}_n, \quad (24)$$

where $\ell \to \infty$ and $\boldsymbol{a}_i$'s are the coordinates of $\boldsymbol{p}^0$ in the eigenspace. It is well-known that the dominant eigenvalue λ_0 is related to their limit positions, whereas the sub-dominant eigenvalues $\lambda_1, \cdots, \lambda_d$ are related to the first order derivatives in the d-dimension space. The rest of eigenvalues must be strictly less than these eigenvalues to guarantee the convergence (see [20]).
In a surface scheme analysis, extraordinary vertices are the only kinds of special cases that we have to consider. However, solid scheme analysis involves not only the analysis of extraordinary vertices, but also that of extraordinary edges. Unlike the relatively simple surface cases, both of the extraordinary cases in solid schemes lack planar symmetry (in general). This situation prohibits a direct application of spectral analysis techniques such as the discrete Fourier transform, which is often employed in eigenanalysis to compute eigenvalues and eigenvectors of the subdivision matrix symbolically (see [6]).
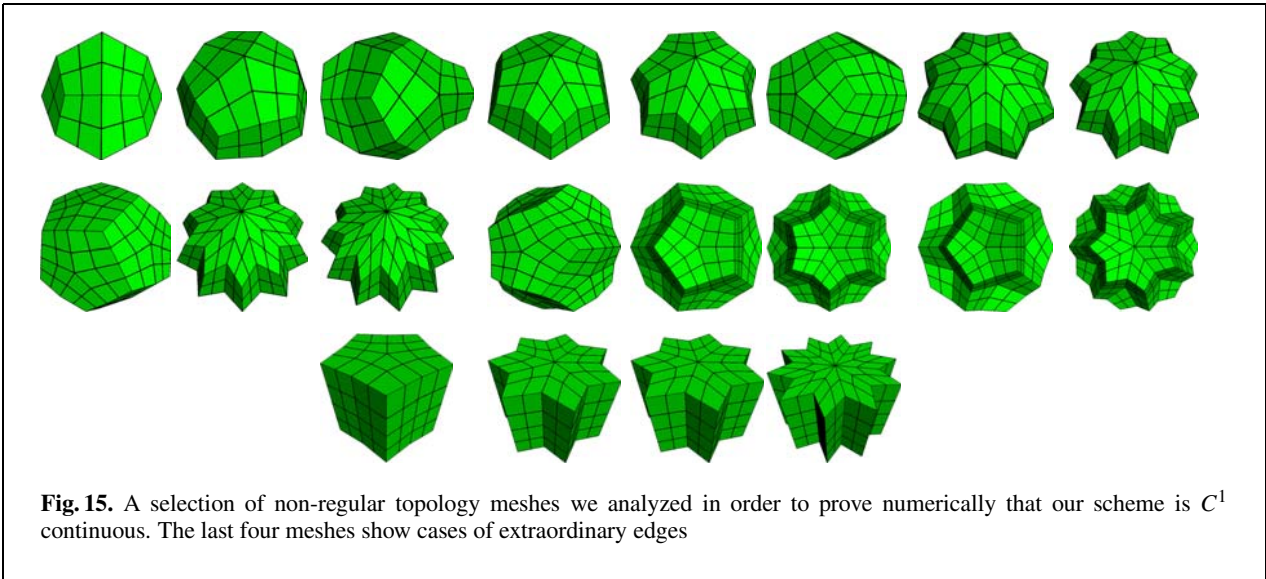
Fig. 15. A selection of non-regular topology meshes we analyzed in order to prove numerically that our scheme is C^1 continuous. The last four meshes show cases of extraordinary edges

To overcome these difficulties, we have computed eigenvalues and eigenvectors of the subdivision matrix around the extraordinary cases numerically. Obviously, it is not possible to acquire the proof of the general cases in this way. We have selected over 20 extraordinary cases and have analyzed their eigenvalues and eigenvectors to verify the necessary conditions for the convergence of the scheme around the extraordinary cases. For the selected cases we have also have numerically performed characteristic map analysis [17], which is a well-understood technique for surface subdivision analysis. Reif [17] proved that the regularity (one-to-one and non-singular aspect) of the characteristic maps guarantees the C^1 continuity of a subdivision scheme around extraordinary cases in the limit.

For all the cases of extraordinary topology we examined, a local subdivision matrix S of the scheme satisfied an eigenvalue property of

$$\lambda_0 = 1 \gneqq \lambda_1 \geq \lambda_2 \geq \lambda_3 \gneqq \lambda_4, \dots, \lambda_n, \qquad (25)$$

where λ_i's are the eigenvalues of S in decreasing order. This property is also in accordance with the empirical results by Bajaj et al. [1] The eigenvalues of the selected cases are listed in Tables 1 and 2. The weight for the scheme is given as $w = \frac{1}{16}$ in all cases. It is important to mention that we have three subdominant eigenvalues that are strictly less than 1. In addition, Fig. 15 shows the control nets of the characteristic map from each of the subdominant eigenvectors. It is important to note that

Table 1. Eigenvalues for a selection of the extraordinary vertex cases

Valence	λ_0	λ_1	λ_2	λ_3	λ_4
5	1	0.499420	0.454695	0.454695	0.352565
7	1	0.500000	0.484286	0.484286	0.404687
8	1	0.465107	0.458056	0.455922	0.447308
8	1	0.500000	0.470715	0.470715	0.461119
9	1	0.500000	0.495818	0.495818	0.439426
10	1	0.475022	0.475022	0.459298	0.445021
10	1	0.500000	0.500000	0.500000	0.432938
11	1	0.500000	0.494283	0.494283	0.470715
12	1	0.459298	0.459298	0.459298	0.445021
12	1	0.500000	0.490543	0.490543	0.484286
13	1	0.500000	0.498844	0.498844	0.472785
14	1	0.470722	0.470722	0.459298	0.445021
17	1	0.474916	0.462847	0.462847	0.448390
20	1	0.474916	0.463449	0.463449	0.452999
22	1	0.470597	0.459411	0.459411	0.457600
23	1	0.474916	0.465931	0.465931	0.465700

Table 2. Eigenvalues for a selection of the extraordinary edge cases

Face number	λ_0	λ_1	λ_2	λ_3	λ_4
5	1	0.5	0.484286	0.484286	0.404687
6	1	0.5	0.470715	0.470715	0.461119
7	1	0.5	0.495818	0.495818	0.439426
10	1	0.5	0.490543	0.490543	0.484286

Cell	Face
List of **Vertices** List of **FaceUse**s	List of **EdgeUse**s Cell neighbor1, neighbor2
Edge	**Vertex**
Vertex endPt1, endPt2 List of **Cell**s List of **Face**s	List of **Cell**s List of **Face**s List of **Edge**s
FaceUse	**EdgeUse**
Face face *flag* orientation	**Edge** edge *flag* orientation

Fig. 16. Information stored in our version of the radial-edge data structure

although the extraordinary vertex cases we have chosen for each valence do not represent all the possible configurations, they constitute a broad selection of the topologies one might encounter in practice.

6 Experimental results and implementation issues

As can be seen in Figs. 17–23, we have applied our new subdivision scheme over a wide variety of geometries and topologies. All the examples shown in the figures exhibit some kind of irregularities. We implemented our subdivision scheme in C++ on a desktop PC equipped with a 2.2 GHz CPU and 1.0 GB RAM. Rendering was done with OpenGL.

6.1 Data structures

In our implementation of the subdivision algorithm, we employ a simplified version of the radial-edge data structure [15, 19], which is a generalization of the winged-edge data structure to arbitrary manifolds. Our implementation consists of four lists to store the cells, faces, edges and vertices. Each topological entity (cell, face, edge, or vertex) contains several short lists that represent its local topological neighborhood. For instance, in our implementation, a face object consists of an ordered list of directed edges and a pair of pointers to the cells

that share the face. Figure 16 illustrates the information stored for each type of topological entity appearing in the data structure. A *use* object is employed to store the orientation of an edge (or face) with respect to the face (or cell) in which it appears. Auxiliary information pertaining to the subdivision scheme is also stored, including a flag indicating whether an entity is in the interior or on the boundary, the coordinate positions of the vertices, etc.

7 Conclusions and future work

We have proposed, derived and analyzed the first subdivision solid algorithm based on the interpolation of arbitrary hexahedral meshes. Our subdivision solid scheme is based on the well-known Lagrange polynomials and has provable convergence and continuity properties. Since our algorithm is interpolatory and uses hexahedral meshes for shape representation, it can be easily incorporated into a wide variety of solid modeling applications. We have shown that the scheme is C^1 continuous in the regular case and for many topological configurations. Hence, it can be used to construct and subdivide complex hexahedral meshes of arbitrary geometry and topology without loss of smoothness in the interior.

Future work involving subdivision solid schemes can be divided into two categories: theoretical and practical. We anticipate that volumetric subdivision algorithms will continue to develop and improve in response to strong user demand for algorithms of higher continuity. It is also imperative that a general scheme for analyzing subdivision solid algorithms be developed. We have taken the first steps in this direction. We are also investigating local, adaptive subdivision (refinement) algorithms, which could further broaden their application scope, but which would require significant changes to the subdivision scheme and its data structures. This lack of a local, adaptive refinement algorithm is one of the limitations of our present work and will be investigated aggressively in the future.

We envision applications of subdivision solid schemes in other domains such as volume visualization, data representation and compression, and dynamic simulation. Using the subdivision rules, we will be able to interpolate smoothly values other than geometry, such as mass, damping, stiffness,
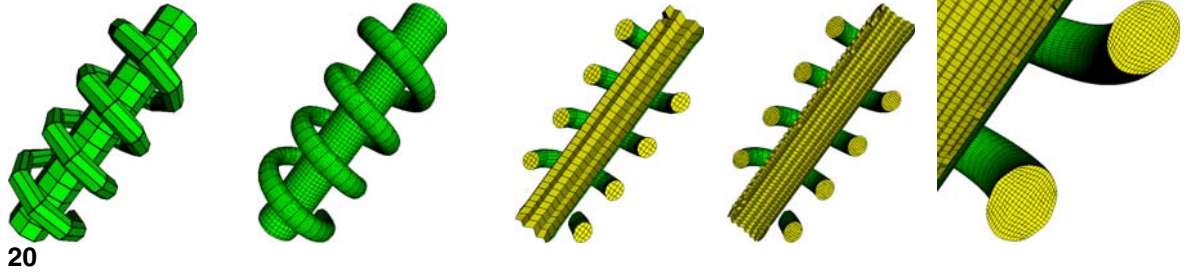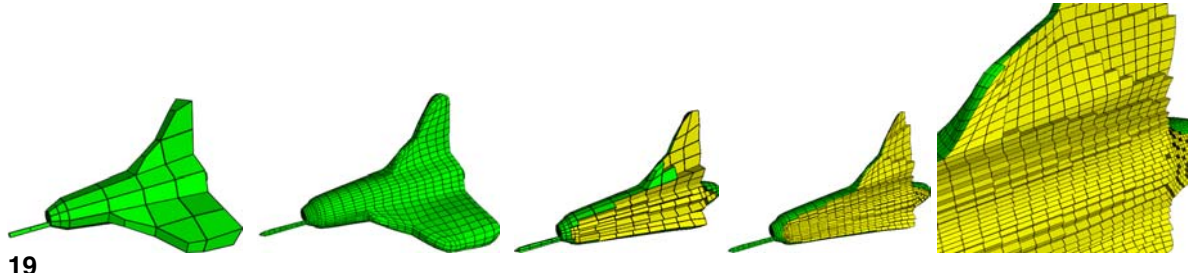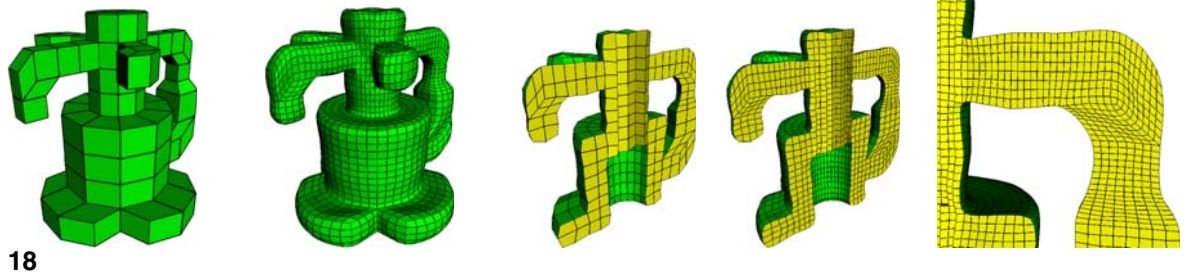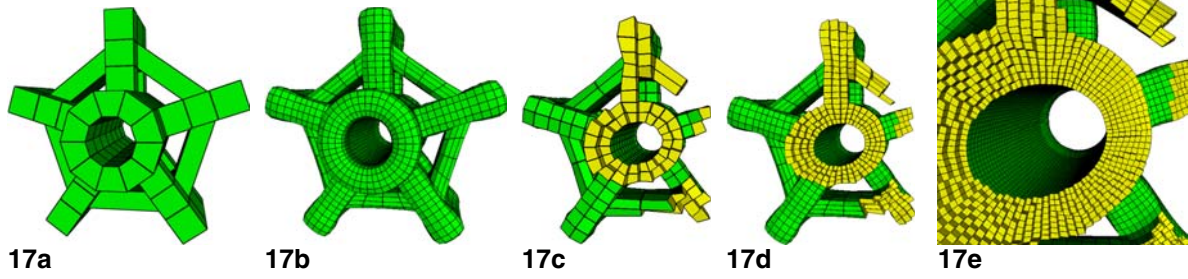
17a   **17b**   **17c**   **17d**   **17e**
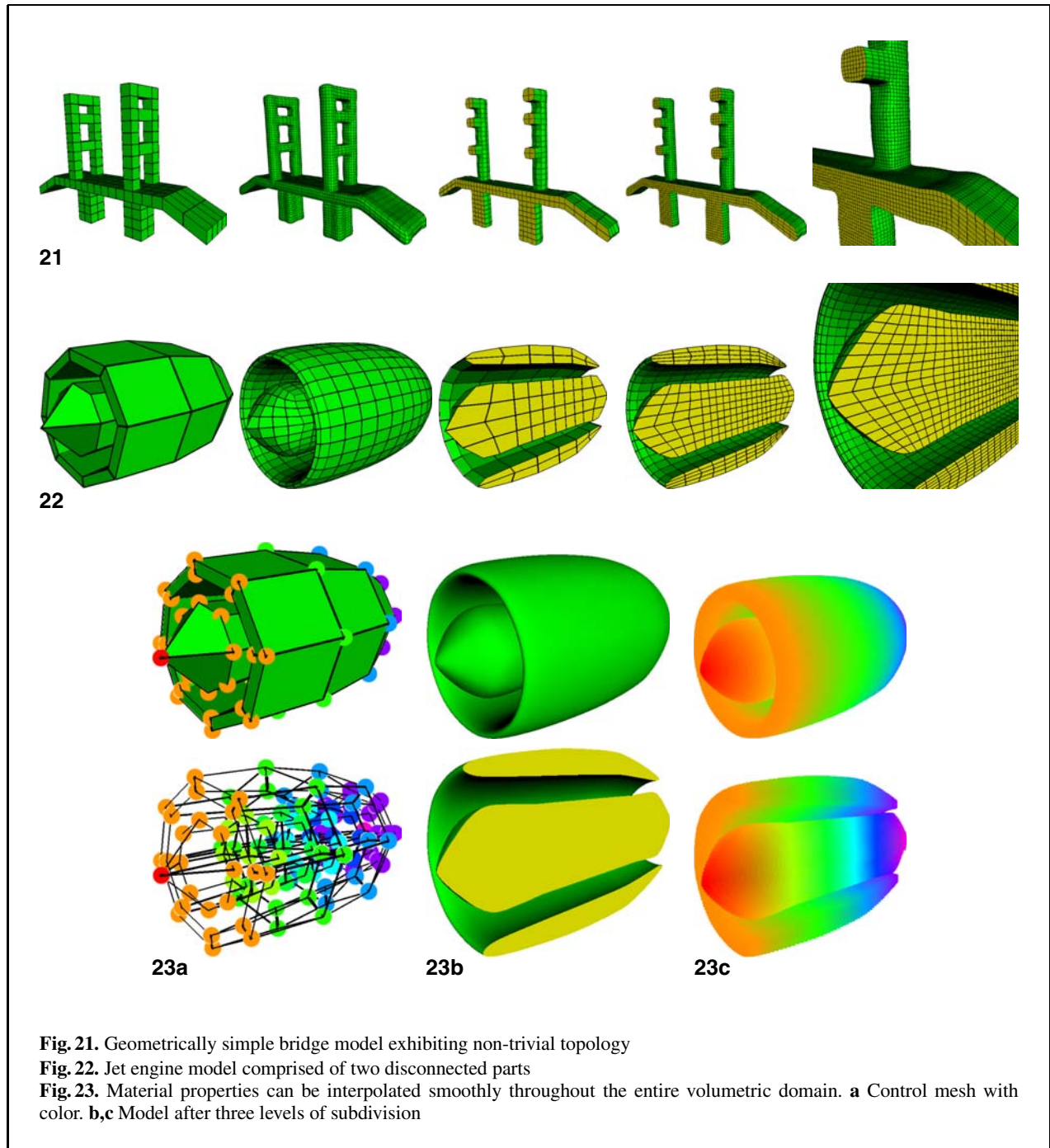
18

19

20

Fig. 17. Topologically complex model featuring many holes and handles. **a** Control mesh. **b** Control mesh after two levels of subdivision. **c,d** A cut-away view of the model after one and two levels of subdivision, respectively. **e** Zoomed view of model after three levels of subdivision

Fig. 18. Mechanical part exhibiting complex topology and extraordinary vertices in the interior

Fig. 19. Small features can be represented and attached to large models without patching

Fig. 20. Disconnected portions of a model can be represented in a single mesh

Fig. 21. Geometrically simple bridge model exhibiting non-trivial topology

Fig. 22. Jet engine model comprised of two disconnected parts

Fig. 23. Material properties can be interpolated smoothly throughout the entire volumetric domain. **a** Control mesh with color. **b,c** Model after three levels of subdivision

etc. As an example, our scheme was used to generate the color map depicted in Fig. 23 by subdividing the red, green and blue color components. We are presently exploring such ideas and are investigating how the unique advantages of subdivision solid algorithms can be exploited in other applications.

References

1. Bajaj C, Shaefer S, Warren J, Xu G (2002) A subdivision scheme for hexahedral meshes. Vis Comput 18(5–6):343–356
2. Bertram M (2002) Biorthogonal wavelets for subdivision volumes. Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, pp 72–82
3. Catmull E, Clark J (1978) Recursively generated B-spline surfaces on arbitrary topological meshes. Comput Aided Des 10:350–355
4. Chang Y, McDonnell KT, Qin H (2002) A new solid subdivision scheme based on Box splines. Proceedings of Seventh ACM Symposium on Solid Modeling and Applications (Solid Modeling 2002), Saarbruecken, Germany, pp 226–233
5. Chang Y, McDonnell KT, Qin H (2003) An interpolatory subdivision for volumetric models over simplicial complexes. Proceedings of the International Conference on Shape Modeling and Applications (SMI 2003), Seoul, Korea, pp 143–152
6. Doo D (1978) A subdivision algorithm for smoothing down irregularly shaped polyhedrons. Proceedings on Interactive Techniques in Computer Aided Design, pp 157–165
7. Dyn N, Hed S, Levin D (1993) Subdivision schemes for surface interpolation. Proceedings of the 1992 Workshop on Computational Geometry. World Scientific, River Edge, NJ, pp 97–118
8. Dyn N, Levin D, Gregory J (1987) A four-point interpolatory subdivision scheme for curve design. Comput Aided Geom Des 4(4):257–268
9. Dyn N, Levin D, Gregory J (1990) A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans Graph 9(2):160–169
10. Dyn N, Micchelli CA (1990) Using parameters to increase smoothness of curves and, surfaces generated by subdivision. Comput Aided Geom Des 7:129–140
11. Linsen L, Pascucci V, Duchaineau MA, Hamann B, Joy KI (2002) Hierarchical representation of time-varying volume data with $\sqrt[4]{2}$ subdivision and quadrilinear B-spline wavelets. Proceedings of Tenth Pacific Conference on Computer Graphics and Applications, pp 346–355
12. MacCracken R, Joy KI (1996) Free-form deformations with lattices of arbitrary topology. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, pp 181–188
13. McDonnell KT, Qin H (2001) FEM-based subdivision solids for dynamic and haptic interaction. Proceedings of Sixth ACM Symposium on Solid Modeling and Applications (Solid Modeling 2001), Ann Arbor, Michigan, pp 312–313
14. McDonnell KT, Qin H, Wlodarczyk RA (2001) Virtual clay: A real-time sculpting system with haptic toolkits. Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, North Carolina, pp 179–190
15. Muuss MJ, Butler LA (1991) Combinatorial solid geometry, boundary representations, and n-manifold geometry. In: Rogers DF, Earnshaw RA (eds) State of the art in computer graphics: visualization and modeling. Springer, Berlin Heidelberg New York, pp 185–223
16. Pascucci V, Bajaj C (2000) Time critical isosurface refinement and smoothing. Proceedings of the 2000 IEEE Symposium on Volume Visualization, pp 33–42
17. Reif U (1995) A unified approach to subdivision algorithms near extraordinary vertices. Comput Aided Geom Des 12:153–174
18. Warren J, Weimer H (2001) Subdivision methods for geometric design: a constructive approach. Kaufmann, San Francisco
19. Weiler KJ (1986) Topological structures for geometric modeling. Dissertation, Rensselaer Polytechnic Institute
20. Zorin D (2000) Smoothness of stationary subdivision on irregular meshes. Constr Approx 16:359–398

Photographs of the authors and their biographies are given on the next page.
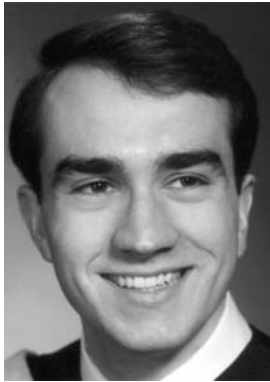
DR. KEVIN T. MCDON-NELL is a research scientist in the Department of Computer Science at Stony Brook University, where he is also a fellow of the Center for Visual Computing (CVC). He received his Ph.D. in computer science at Stony Brook University in Computer Science in 2003, where he also earned his M.S. degree. He graduated *summa cum laude* with a B.S. in both computer science and applied mathematics from Stony Brook in 1998. In 1998 he was awarded both a University Graduate Research Fellowship and a Graduate Assistance in Areas of National Need (GAANN) fellowship. His research interests include physically based modeling, geometric modeling, scientific visualization, and interactive 3D graphics. He is a member of Phi Beta Kappa and the Golden Key National Honor Society.

YU-SUNG CHANG is a Ph.D. student in the Department of Computer Science at Stony Brook University, where he is also a fellow of the Center for Visual Computing (CVC). He earned his B.S. degree in mathematics from Seoul National University, Korea in 1996. He received his M.S. degree in mathematics from the Courant Institute at New York University in 2000. From 1991–1997, he received an honor scholarship and was awarded the Honor for Excellent Graduation in 1996 from Seoul National University. In 2000, he received a Presidential Fellowship from Stony Brook University. His primary research interests include geometric modeling, subdivision analysis, and human-computer interaction in virtual environments. He is a member of ACM, SIAM and MAA.

DR. HONG QIN is an associate professor of computer science at the State University of New York at Stony Brook, where he is also a member of the SUN-YSB Center for Visual Computing. He received his B.S. degree (1986) and his M.S. degree (1989) in computer science from Peking University in Beijing, P.R. China. He received his Ph.D. degree (1995) in computer science from the University of Toronto. From 1989–90 he was a research scientist at the North-China Institute of Computing Technologies. From 1990–1991 he was a Ph.D. candidate in computer science at the University of North Carolina at Chapel Hill. From 1996–1997, he was an assistant professor of computer & information science & engineering at the University of Florida. He received the Honor Student Award from 1983–85 and the Best Graduate Award in 1986 from Peking University. During his years at the University of Toronto, he received an Open Doctoral Fellowship. In 1997, Dr. Qin was awarded the NSF CAREER Award from the National Science Foundation (NSF), and, in September, 2000, was awarded a newly-established NSF Information Technology Research (ITR) grant. In December, 2000, he received a Honda Initiation Grant Award, and, in April, 2001, was selected as an Alfred P. Sloan Research Fellow by the Sloan Foundation. He is a member of ACM, IEEE, SIAM, and Eurographics.