Deformable surface modeling based on dual subdivision *

WANG Huawei[1] **, SUN Hanqiu[2] and QIN Kaihuai[1]

(1. Department of Computer Science & Technology, Tsinghua University, Beijing 100084, China; 2. Department of Computer Science & Engineering, Chinese University of Hong Kong, Hong Kong, China)

Received February 20, 2004; revised August 20, 2004

Abstract Based on dual Doo-Sabin subdivision and the corresponding parameterization, a modeling technique of deformable surfaces is presented in this paper. In the proposed model, all the dynamic parameters are computed in a unified way for both non-defective and defective subdivision matrices, and central differences are used to discretize the Lagrangian dynamics equation instead of backward differences. Moreover, a local scheme is developed to solve the dynamics equation approximately, thus the order of the linear equation is reduced greatly. Therefore, the proposed model is more efficient and faster than the existing dynamic models. It can be used for deformable surface design, interactive surface editing, medical imaging and simulation.

Keywords: deformable model; subdivision surface; dynamics; Doo-Sabin surface.

Free-form surfaces, especially non-uniform rational B-splines (NURBS), are widely used in computer aided geometric design and computer-aided design/manufacturing. However, usually a surface of arbitrary topology cannot be represented by a single NURBS. Moreover, it is very difficult to set the knot spacings, manipulate the control vertices or modify the weights to meet some design requests using NURBS. Consequently, subdivision surfaces appear to uniformly model any complex surfaces of arbitrary topology[1]. However, interactive methods for modifying subdivision surfaces need to be developed further. Attention has been paid to physics-based dynamic subdivision surfaces in recent years[2—5], because dynamic subdivision surfaces allow users to edit the shapes of limit surfaces by applying forces at the desired locations of the control meshes, which seems very intuitive and natural just as modeling shapes in clay.

Qin et al. introduced the "physical quantity" into dynamic Catmull-Clark surfaces in 1998, which were successfully applied to the visualization of medical data[2]. In 2000, Mandal et al. gave a finite-element-method-based dynamic framework for subdivision surfaces and dealt with the modified butterfly and Catmull-Clark subdivision schemes[3]. However, they computed the dynamic parameters, such as the mass, damping and stiffness matrices, by subdividing the control mesh recursively, and obtained only the approximate results. Using parameterizations for subdivision surfaces of arbitrary topologies including Catmull-Clark surfaces and Loop surfaces[6,7], Qin et al. showed that it was desirable to evaluate all the dynamic parameters exactly without subdividing the control mesh recursively[4,5]. However, their method is not applicable for general subdivision matrices.

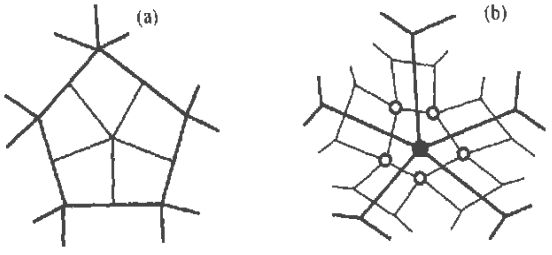

Fig. 1. Primal and dual subdivision schemes. (a) Primal scheme (face split); (b) dual scheme (vertex split).

In general, the existing subdivision schemes can be classified into two types: primal and dual, i.e. face split and vertex split[1]. As shown in Fig. 1, in the former case, N new faces are created for each N-sided face, while in the latter case, new vertices are created for each old vertex, one for each face containing the vertex. However, the subdivision schemes mentioned above are all primal, thus it is desirable to develop dynamic surface models based on dual subdivision. Doo-Sabin scheme[8], for instance, is a typi-

* Supported by the RGC research grant of Hong Kong (Grant Nos. 4356/02E and 4181/03E) and the National Natural Science Foundation of China (Grant No. 60273013)

** To whom correspondence should be addressed. E-mail: whw98@mails.tsinghua.edu.cn

cal dual subdivision scheme. Because the subdivision matrix of Doo-Sabin scheme may be defective, in contrast to the non-defective cases discussed in previous work, it is challenging to compute the dynamic parameters for physics-based Doo-Sabin surfaces. Recently, Wang and Qin gave a precise evaluation scheme for Doo-Sabin surfaces such that the values and derivatives of Doo-Sabin surfaces can be computed easily in any parameter positions[9]. Using this evaluation scheme, we will present a novel deformable surface model based on Doo-Sabin subdivision, and extend the dynamic subdivision surfaces to the dual setting.

1 Uniform Doo-Sabin subdivision surfaces

The uniform Doo-Sabin surface is designed by generalizing the uniform biquadratic B-spline surface to meshes of arbitrary topology. The surface is defined as the limit of the control mesh which becomes finer and finer by being subdivided recursively. At each step new vertices are introduced and new faces (of type F, type E, and type V, respectively) are created for each face, edge and vertex of the previous polyhedron (see Fig. 2(a)). The well-known rules of Doo-Sabin subdivision are given as[8] (see Fig. 2 (b)):

$$\bar{\boldsymbol{p}}_i = \frac{1}{4}\boldsymbol{p}_i + \frac{1}{4N}\sum_{j=0}^{N-1}\left(3 + 2\cos\frac{2\pi\,|\,i-j\,|}{N}\right)\boldsymbol{p}_j,$$

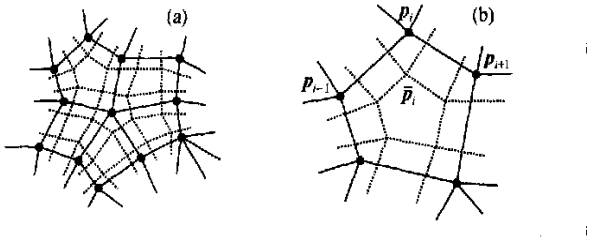

Fig. 2. Doo-Sabin subdivision of control meshes of arbitrary topology. (a) Subdividing an arbitrary mesh; (b) local structure for computing new vertices.

where $N \geqslant 3$ is the number of the edges of the face $\boldsymbol{p}_0\boldsymbol{p}_1\cdots\boldsymbol{p}_{N-1}$. After one subdivision step, each vertex of the new polyhedron will have valence 4, and the number of non-quadrilateral faces will remain constant. In the refinement process, the extraordinary points are at the centers of the N-sided faces with $N \neq 4$. After one more subdivision step, all extraordinary points are isolated. Thus, we generally assume that all the extraordinary points are isolated in the initial polyhedron so that all vertices have valence 4 and no two faces with $N \neq 4$ sides share a common vertex.

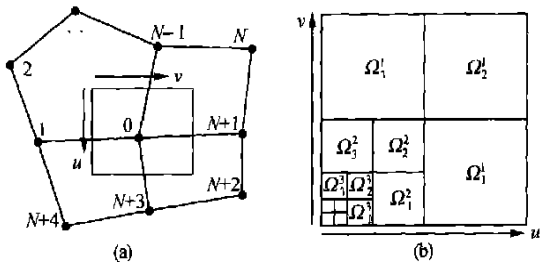

Fig. 3. Parameterization for a surface patch. (a) Local structure defining a patch; (b) infinite partition of the parameter region Ω.

A Doo-Sabin surface is divided naturally into many little surface patches[9], which have a one-to-one relationship with the vertices, in contrast to the one-to-one relationship between the surface patches and the polyhedral faces for primal subdivision surfaces. Each patch is defined by only those faces containing the corresponding vertex, as shown in Fig. 3 (a). Note that the valence of each vertex equals 4 and all the extraordinary points are isolated, then there are three four-sided faces in the local structure and the fourth face is supposed to be an N-sided face. Obviously the surface patch reduces to a uniform biquadratic B-spline surface when N is equal to 4. It is shown in Ref. [9] that the surface patch can be parameterized as a function $\boldsymbol{s}(u, v)$, which is defined on $\Omega = [0, 1] \times [0, 1]$. In detail,

$$\boldsymbol{s}^{\mathrm{T}}(u, v) = \boldsymbol{J}^{*}\boldsymbol{C}_s, \quad (u, v) \in \Omega,$$

where

$$\boldsymbol{C}_s^{\mathrm{T}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_{N+4}),$$

$$\boldsymbol{J}^{*}\Big|_{\Omega_k^n} = \boldsymbol{b}^{\mathrm{T}}(\boldsymbol{t}_{k,n}(u, v))\boldsymbol{X}_k\Sigma^{n-1}\boldsymbol{V}^{-1},$$

$$k = 1, 2, 3, \quad n = 1, 2, \cdots,$$

where $\boldsymbol{X}_k = \boldsymbol{P}_k\bar{\boldsymbol{A}}\boldsymbol{V}$, $(\Sigma, \boldsymbol{V})$ is the eigenstructure of the subdivision matrix $\boldsymbol{A}$, and $\boldsymbol{b}(u, v)$ are uniform biquadratic B-spline basis functions. The formulae of all matrices $(\boldsymbol{A}, \Sigma, \boldsymbol{V}, \bar{\boldsymbol{A}}, \boldsymbol{P}_k)$, basic functions $\boldsymbol{b}(u, v)$, subdomains Ω_k^n and transformations $\boldsymbol{t}_{k,n}(u, v)$ can be found in Ref. [9]. $\{\Omega_k^n,\ k=1, 2, 3,\ n=1, 2, \cdots\}$ constitute a partition of Ω, as shown in Fig. 3 (b). Note that the subdivision matrix $\boldsymbol{A}$ is non-defective for $N \leqslant 4$ but defective for $N > 4$, thus Σ may be a Jordan canonical form, in contrast to the simple diagonal matrices in the previous work. For any N, we rewrite Σ as $\Sigma = \Lambda + \boldsymbol{U}$, where

$$\Lambda = \mathrm{diag}\left(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{16}, \frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \cdots, \frac{1}{4}\right)$$

$$\triangleq \mathrm{diag}(\lambda_0, \lambda_1, \lambda_2, \cdots, \lambda_{N+4}),$$

$$\boldsymbol{U}=\begin{cases}\mathrm{diag}(0,0,\cdots,0,\Delta(0),\Delta(0)), & \text{if } N>4,\\ \boldsymbol{0}, & \text{otherwise,}\end{cases}$$

$$\text{and } \Delta(\lambda)=\begin{pmatrix}\lambda & 1\\ 0 & \lambda\end{pmatrix}.$$

If there are m vertices and d faces in the initial control mesh, then the i-th limit surface patch can be rewritten as

$$\boldsymbol{s}_i^{\mathrm{T}}=\boldsymbol{J}_i\boldsymbol{C}_{s_i}=\boldsymbol{J}^{*}\boldsymbol{R}_i\boldsymbol{C}_0,$$

where $\boldsymbol{C}_0^{\mathrm{T}}=(\boldsymbol{p}_0,\boldsymbol{p}_1,\cdots,\boldsymbol{p}_{m-1})$ contains all m control vertices, and $\boldsymbol{R}_i$ is an $(N+5)\times m$ picking matrix (each row is filled with zeros except for a one in a certain column). Note that $\boldsymbol{J}^{*}$ only depends on the valence N.

2 Deformable model based on Doo-Sabin subdivision

2.1 Decomposition of forces in dynamics equation

If we regard the control vertices $\boldsymbol{C}_0$ as a function related to the time t in a pure physical system, then the Lagrangian dynamics equation is satisfied:

$$\boldsymbol{M}\ddot{\boldsymbol{C}}_0+\boldsymbol{D}\dot{\boldsymbol{C}}_0+\boldsymbol{K}\boldsymbol{C}_0=\boldsymbol{F}_{\mathrm{p}},$$

where $\boldsymbol{M}$, $\boldsymbol{D}$ and $\boldsymbol{K}$ are the mass, damping and stiffness matrices, respectively, and $\boldsymbol{F}_{\mathrm{p}}$ is the generalized force vector. Let μ be the density of mass, γ be the coefficient of damping, then the mass matrix and the damping matrix are

$$\boldsymbol{M}=\iint\mu\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}\,\mathrm{d}u\,\mathrm{d}v,\quad \boldsymbol{D}=\iint\gamma\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}\,\mathrm{d}u\,\mathrm{d}v,$$

where $\boldsymbol{J}=\sum_{i=1}^{d}\boldsymbol{J}^{*}\boldsymbol{R}_i$. From a thin-plate-under-tension energy model, the stiffness matrix can be computed by[2,10]

$$\boldsymbol{K}=\iint(\alpha_{11}\boldsymbol{J}_u^{\mathrm{T}}\boldsymbol{J}_u+\alpha_{22}\boldsymbol{J}_v^{\mathrm{T}}\boldsymbol{J}_v+\beta_{11}\boldsymbol{J}_{uu}^{\mathrm{T}}\boldsymbol{J}_{uu}+\beta_{12}\boldsymbol{J}_{uv}^{\mathrm{T}}\boldsymbol{J}_{uv}+\beta_{22}\boldsymbol{J}_{vv}^{\mathrm{T}}\boldsymbol{J}_{vv})\,\mathrm{d}u\,\mathrm{d}v,$$

where α_{ii} and β_{ij} are the characteristic springiness coefficients.

Suppose that the surface is stable before being deformed, i.e. the surface keeps still, then $\ddot{\boldsymbol{C}}_0=0$ and $\dot{\boldsymbol{C}}_0=0$. Hence the dynamics equation becomes $\boldsymbol{K}\boldsymbol{C}_0=\boldsymbol{F}_{\mathrm{p}}$. That is to say, there are initial forces $\boldsymbol{F}_{\mathrm{s}}=\boldsymbol{K}\boldsymbol{C}_0(0)$, called static forces, initially applied on the surface such that the surface has a stable initial shape. The static forces are assumed to be constants, only depending on the initial Doo-Sabin surface. On the other hand, in order to deform the surface, we apply external forces $\boldsymbol{F}_{\mathrm{e}}$ on the surface. Therefore, the forces $\boldsymbol{F}_{\mathrm{p}}$ are divided into two parts: static forces $\boldsymbol{F}_{\mathrm{s}}$ and external forces $\boldsymbol{F}_{\mathrm{e}}$, and accordingly the Lagrangian dynamics equation can be rewritten as

$$\boldsymbol{M}\ddot{\boldsymbol{C}}_0+\boldsymbol{D}\dot{\boldsymbol{C}}_0+\boldsymbol{K}\boldsymbol{C}_0=\boldsymbol{F}_{\mathrm{s}}+\boldsymbol{F}_{\mathrm{e}},\qquad(1)$$

where $\boldsymbol{F}_{\mathrm{e}}$ is given as:

$$\boldsymbol{F}_{\mathrm{e}}=\iint\boldsymbol{J}^{\mathrm{T}}\boldsymbol{f}^{\mathrm{T}}\,\mathrm{d}u\,\mathrm{d}v.$$

When an external force $\boldsymbol{f}$ is applied at some control vertex, we think it is applied on the surface patch corresponding to the vertex. If the control mesh is subdivided once, the vertex, whose valence is denoted by l, is split into l new vertices (see Fig. 1), and the surface patch is divided into l less patches at the same time. Let each new vertex have a force $\boldsymbol{f}$, then we can easily find that the physical systems are equivalent before and after the subdivision step, that is, the force on the limit surface is fixed everywhere in the subdivision process. In contrast, for a primal subdivision, a force applied at some vertex should be taken into account for all faces containing the vertex, which means that the force is applied on all limit surface patches corresponding to these faces. From another point of view, each surface patch depends on all control vertices within its one-neighborhood in the aspect of applying forces. Such complicated relations between the surface patches and control vertices make it difficult to assign the forces equivalently for the resulting mesh if the control mesh is subdivided. Thus, dual subdivision, considered in this paper, is more convenient to be used in the interactive modeling process with both applications of forces and subdivisions of the meshes.

2.2 Evaluating dynamic parameters

In this subsection, we will present the approaches for computing all the dynamic parameters and begin with the mass matrix $\boldsymbol{M}$. By subdividing the control mesh several times, we can suppose that the density of mass μ is constant in each face for the sake of simpleness, then

$$\boldsymbol{M}=\sum_{i=1}^{d}\mu_i\boldsymbol{R}_i^{\mathrm{T}}\left(\iint\boldsymbol{J}^{*\mathrm{T}}\boldsymbol{J}^{*}\,\mathrm{d}u\,\mathrm{d}v\right)\boldsymbol{R}_i.$$

Using the definition of $\boldsymbol{J}^{*}$, we can obtain

$$\iint\boldsymbol{J}^{*\mathrm{T}}\boldsymbol{J}^{*}\,\mathrm{d}u\,\mathrm{d}v=\sum_{k=1}^{3}\sum_{n=1}^{\infty}\iint_{\Omega_k^n}\boldsymbol{J}^{*\mathrm{T}}\boldsymbol{J}^{*}\,\mathrm{d}u\,\mathrm{d}v$$

$$=\sum_{k=1}^{3}\sum_{n=1}^{\infty}\frac{1}{4^n}\boldsymbol{V}^{-\mathrm{T}}(\Sigma^{T})^{n-1}\boldsymbol{X}_k^{\mathrm{T}}$$

$$\cdot\left(\iint_{\Omega}\boldsymbol{b}(u,v)\boldsymbol{b}^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v\right)\boldsymbol{X}_k\Sigma^{n-1}\boldsymbol{V}^{-1}.$$

Every element in $\boldsymbol{b}(u,v)$ is a polynomial on Ω, thus $\iint_{\Omega}\boldsymbol{b}(u,v)\boldsymbol{b}^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v$ can be computed directly.

Let $\boldsymbol{Z}_k=\boldsymbol{X}_k^{\mathrm{T}}\left(\iint_{\Omega}\boldsymbol{b}(u,v)\boldsymbol{b}^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v\right)\boldsymbol{X}_k$, and

$$\boldsymbol{Q}_k=\sum_{n=1}^{\infty}4^{-n}(\Sigma^T)^{n-1}\boldsymbol{Z}_k\Sigma^{n-1}.$$

Since $\Sigma^{n-1}=\Lambda^{n-1}+(n-1)4^{-n+2}\boldsymbol{U}$, and $\boldsymbol{Z}_k$ is a symmetrical matrix, we can obtain

$$\boldsymbol{Q}_k=\boldsymbol{Q}_k^1+\boldsymbol{Q}_k^2+(\boldsymbol{Q}_k^2)^{\mathrm{T}}+\boldsymbol{Q}_k^3,$$

where

$$\boldsymbol{Q}_k^1=\sum_{n=1}^{\infty}4^{-n}\Lambda^{n-1}\boldsymbol{Z}_k\Lambda^{n-1},$$

$$\boldsymbol{Q}_k^2=\sum_{n=1}^{\infty}4^{-n}\Lambda^{n-1}\boldsymbol{Z}_k(n-1)4^{-n+2}\boldsymbol{U},$$

and

$$\boldsymbol{Q}_k^3=\sum_{n=1}^{\infty}4^{-n}(n-1)4^{-n+2}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{Z}_k(n-1)4^{-n+2}\boldsymbol{U}.$$

Each element of $\boldsymbol{Q}_k^1$ is a power series, and accordingly can be obtained immediately:

$$(\boldsymbol{Q}_k^1)_{ij}=\sum_{n=1}^{\infty}\frac{1}{4^n}(\lambda_i\lambda_j)^{n-1}(\boldsymbol{Z}_k)_{ij}=\frac{1}{4-\lambda_i\lambda_j}(\boldsymbol{Z}_k)_{ij}.$$

Note that

$$\boldsymbol{Q}_k^2=\sum_{n=1}^{\infty}(n-1)4^{-2n+2}\Lambda^{n-1}\boldsymbol{Z}_k\boldsymbol{U}$$
$$=\mathrm{diag}(\sigma_0,\sigma_1,\cdots,\sigma_{N+4})\boldsymbol{Z}_k\boldsymbol{U},$$

where

$$\sigma_i=\sum_{n=1}^{\infty}(n-1)4^{-2n+2}\lambda_i^{n-1}$$

is obviously a convergent series and can be computed easily by program, $i=0,1,\cdots,N+4$. Similarly,

$$\boldsymbol{Q}_k^3=\boldsymbol{U}^{\mathrm{T}}\boldsymbol{Z}_k\boldsymbol{U}\left(\sum_{n=1}^{\infty}(n-1)^2 4^{-3n+4}\right)$$

can be calculated. As a result, we obtain

$$\iint\boldsymbol{J}^{*\mathrm{T}}\boldsymbol{J}^{*}\mathrm{d}u\mathrm{d}v=\sum_{k=1}^{3}\boldsymbol{V}^{-\mathrm{T}}\boldsymbol{Q}_k\boldsymbol{V}^{-1},$$

as well as the mass matrix $\boldsymbol{M}$. Then, the damping matrix

$$\boldsymbol{D}=\sum_{i=1}^{d}\gamma_i\boldsymbol{R}_i^{\mathrm{T}}\left(\iint\boldsymbol{J}^{*\mathrm{T}}\boldsymbol{J}^{*}\mathrm{d}u\mathrm{d}v\right)\boldsymbol{R}_i$$

can also be obtained if the damping coefficient γ is constant in each face. In the same way, we can obtain the force vector

$$\boldsymbol{F}_{\mathrm{e}}=\sum_{i=1}^{d}\boldsymbol{R}_i^{\mathrm{T}}\left(\iint\boldsymbol{J}^{*\mathrm{T}}\mathrm{d}u\mathrm{d}v\right)\boldsymbol{f}_i^{\mathrm{T}}$$

by computing each integral

$$\iint\boldsymbol{J}^{*\mathrm{T}}\mathrm{d}u\mathrm{d}v$$
$$=\sum_{k=1}^{3}\boldsymbol{V}^{-\mathrm{T}}\left(\sum_{n=1}^{\infty}(2^{-n}\Lambda^{n-1}+(n-1)2^{-3n+4}\boldsymbol{U}^{\mathrm{T}})\right)$$
$$\cdot\boldsymbol{X}_k^{\mathrm{T}}\left(\iint_{\Omega}\boldsymbol{b}(u,v)\mathrm{d}u\mathrm{d}v\right).$$

Next, we will investigate how to compute the stiffness matrix

$$\boldsymbol{K}=\sum_{i=1}^{d}\boldsymbol{R}_i^{\mathrm{T}}\left(\iint\alpha_{11}^i\boldsymbol{J}_u^{*\mathrm{T}}\boldsymbol{J}_u^{*}\mathrm{d}u\mathrm{d}v+\iint\alpha_{22}^i\boldsymbol{J}_v^{*\mathrm{T}}\boldsymbol{J}_v^{*}\mathrm{d}u\mathrm{d}v\right.$$
$$+\iint\beta_{11}^i\boldsymbol{J}_{uu}^{*\mathrm{T}}\boldsymbol{J}_{uu}^{*}\mathrm{d}u\mathrm{d}v+\iint\beta_{12}^i\boldsymbol{J}_{uv}^{*\mathrm{T}}\boldsymbol{J}_{uv}^{*}\mathrm{d}u\mathrm{d}v$$
$$\left.+\iint\beta_{22}^i\boldsymbol{J}_{vv}^{*\mathrm{T}}\boldsymbol{J}_{vv}^{*}\mathrm{d}u\mathrm{d}v\right)\boldsymbol{R}_i,$$

which contains the first order or second order derivatives in each integral. Considering the C^1 continuity of Doo-Sabin surfaces, we need to deal with the derivatives of first order and second order, respectively. For the integrals containing first order derivatives, we assume the coefficients α_{jj}^i to be constant, and accordingly we need to evaluate the integrals $\iint\boldsymbol{J}_u^{*\mathrm{T}}\boldsymbol{J}_u^{*}\mathrm{d}u\mathrm{d}v$ and $\iint\boldsymbol{J}_v^{*\mathrm{T}}\boldsymbol{J}_v^{*}\mathrm{d}u\mathrm{d}v$. Using the formulae of derivatives[6,9]:

$$\frac{\partial^{i+j}}{\partial u^i\partial v^j}\boldsymbol{J}^{*}\Big|_{\Omega_k^n}=2^{(i+j)n}\boldsymbol{b}_{u^iv^j}^{\mathrm{T}}(\boldsymbol{t}_{k,n}(u,v))\boldsymbol{X}_k\Sigma^{n-1}\boldsymbol{V}^{-1},$$
$$\forall k,n,i,j,$$

we can obtain

$$\iint\boldsymbol{J}_u^{*\mathrm{T}}\boldsymbol{J}_u^{*}\mathrm{d}u\mathrm{d}v=\sum_{k=1}^{3}\sum_{n=1}^{\infty}4^n\frac{1}{4^n}\boldsymbol{V}^{-\mathrm{T}}(\Sigma^{\mathrm{T}})^{n-1}\boldsymbol{X}_k^{\mathrm{T}}$$
$$\cdot\left(\iint_{\Omega}\boldsymbol{b}_u(u,v)\boldsymbol{b}_u^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v\right)\boldsymbol{X}_k\Sigma^{n-1}\boldsymbol{V}^{-1}.$$

If we rewrite Λ as $\Lambda=\Lambda_1+\Lambda_2$, where

$$\Lambda_1=\mathrm{diag}(1,0,0,\cdots,0)$$

and

$$\Lambda_2=\mathrm{diag}\left(0,\frac{1}{2},\frac{1}{2},\frac{1}{16},\frac{1}{8},\frac{1}{8},\frac{1}{4},\frac{1}{4},\cdots,\frac{1}{4}\right),$$

and denote the first column of $\boldsymbol{V}$ by $\boldsymbol{v}_0=(a,a,\cdots,a)^{\mathrm{T}}$, then $\boldsymbol{V}\Lambda_1=(\alpha,\alpha,\cdots,\alpha)^{\mathrm{T}}$, where $\alpha=(a,0,0,\cdots,0)^{\mathrm{T}}$. Because the sum of elements in each row of $\boldsymbol{P}_k$ or $\bar{\boldsymbol{A}}$ is 1, we have $\bar{\boldsymbol{A}}\boldsymbol{V}\Lambda_1=(\alpha,\alpha,\cdots,\alpha)^{\mathrm{T}}$, and then $\boldsymbol{P}_k\bar{\boldsymbol{A}}\boldsymbol{V}\Lambda_1=(\alpha,\alpha,\cdots,\alpha)^{\mathrm{T}}$. It is followed that

$$\boldsymbol{b}^{\mathrm{T}}(u,v)\boldsymbol{P}_k\bar{\boldsymbol{A}}\boldsymbol{V}\Lambda_1=\sum_{i=0}^{8}(\boldsymbol{b}(u,v))_i\alpha^{\mathrm{T}}=\alpha^{\mathrm{T}},$$

so

$$\boldsymbol{b}_u^{\mathrm{T}}(u,v)\boldsymbol{P}_k\bar{\boldsymbol{A}}\boldsymbol{V}\Lambda_1=\frac{\partial}{\partial u}(\alpha^{\mathrm{T}})=0$$
$$=\boldsymbol{b}_u^{\mathrm{T}}(u,v)\boldsymbol{X}_k\Lambda_1.$$

In addition, $\Sigma^{n-1} = \Lambda_1 + \Lambda_2^{n-1} + (n-1)4^{-n+2}\boldsymbol{U}$, thus we have

$$\iint \boldsymbol{J}_u^{*\mathrm{T}} \boldsymbol{J}_u^{*} \mathrm{d}u\mathrm{d}v = \sum_{k=1}^{3}\sum_{n=1}^{\infty} \boldsymbol{V}^{-\mathrm{T}}(\Lambda_2^{n-1} + (n-1)4^{-n+2}\boldsymbol{U}^{\mathrm{T}}) \circ \boldsymbol{Z}_k'(\Lambda_2^{n-1} + (n-1)4^{-n+2}\boldsymbol{U})\boldsymbol{V}^{-1},$$

where

$$\boldsymbol{Z}_k' = \boldsymbol{X}_k^{\mathrm{T}}\left(\iint_{\Omega} \boldsymbol{b}_u(u,v)\boldsymbol{b}_u^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v\right)\boldsymbol{X}_k.$$

Let $\boldsymbol{Q}_k' = \boldsymbol{Q}_k'^1 + \boldsymbol{Q}_k'^2 + (\boldsymbol{Q}_k'^2)^{\mathrm{T}} + \boldsymbol{Q}_k'^3$, where

$$\boldsymbol{Q}_k'^1 = \sum_{n=1}^{\infty} \Lambda_2^{n-1}\boldsymbol{Z}_k'\Lambda_2^{n-1},$$

$$\boldsymbol{Q}_k'^2 = \sum_{n=1}^{\infty} \Lambda_2^{n-1}\boldsymbol{Z}_k'(n-1)4^{-n+2}\boldsymbol{U},$$

$$\boldsymbol{Q}_k'^3 = \sum_{n=1}^{\infty} (n-1)4^{-n+2}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{Z}_k'(n-1)4^{-n+2}\boldsymbol{U},$$

then every element of $\boldsymbol{Q}_k'^1$, $\boldsymbol{Q}_k'^2$ and $\boldsymbol{Q}_k'^3$ is a convergent series, so $\boldsymbol{Q}_k'$ can be computed as above. For instance,

$$(\boldsymbol{Q}_k'^1)_{ij} = \begin{cases} 0, & \text{if } i=0 \text{ or } j=0, \\ \dfrac{1}{1-\lambda_i\lambda_j}(\boldsymbol{Z}_k')_{ij}, & \text{otherwise.} \end{cases}$$

Then, we have obtained

$$\iint \boldsymbol{J}_u^{*\mathrm{T}}\boldsymbol{J}_u^{*}\mathrm{d}u\mathrm{d}v = \sum_{k=1}^{3} \boldsymbol{V}^{-\mathrm{T}}\boldsymbol{Q}_k'\boldsymbol{V}^{-1}.$$

Analogously, $\iint \boldsymbol{J}_v^{*\mathrm{T}}\boldsymbol{J}_v^{*}$ can be calculated.

Note that Doo-Sabin surfaces are not C^2 continuous but only C^1 continuous, and the second order derivatives may diverge near an extraordinary point, so we set the characteristic springiness coefficients β_{ij} to be zero in a very tiny neighborhood of the extraordinary point, similar to Ref. [2], in order to avoid computing certain integrals for special cases. For example, we can set $\beta_{ij}^r(u,v) = b_{ij}^r\varphi(u,v)$, where

$$\varphi(u,v) = \begin{cases} 0, & \text{if } (u,v) \in [0, 2^{-20}] \times [0, 2^{-20}], \\ 1, & \text{otherwise,} \end{cases}$$

then

$$\iint \beta_{11}^i \boldsymbol{J}_{uu}^{*\mathrm{T}}\boldsymbol{J}_{uu}^{*}\mathrm{d}u\mathrm{d}v = b_{11}^i \sum_{k=1}^{3}\sum_{n=1}^{20} 4^{2n}\frac{1}{4^n}\boldsymbol{V}^{-\mathrm{T}}(\Lambda_2^{n-1} + (n-1)4^{-n+2}\boldsymbol{U}^{\mathrm{T}}) \circ \boldsymbol{X}_k^{\mathrm{T}}\left(\iint_{\Omega} \boldsymbol{b}_{uu}(u,v)\boldsymbol{b}_{uu}^{\mathrm{T}}(u,v)\mathrm{d}u\mathrm{d}v\right) \circ \boldsymbol{X}_k(\Lambda_2^{n-1} + (n-1)4^{-n+2}\boldsymbol{U})\boldsymbol{V}^{-1}. \quad (2)$$

Since $\iint_{\Omega} \boldsymbol{b}_{uu}(u,v)\boldsymbol{b}_{uu}^{T}(u,v)\,du\,dv$ can be obtained directly, we can similarly simplify the right-hand part of Eq. (2) and then calculate it. $\iint \beta_{12}^i \boldsymbol{J}_{uv}^{*\mathrm{T}}\boldsymbol{J}_{uv}^{*}\mathrm{d}u\mathrm{d}v$ and $\iint \beta_{22}^i \boldsymbol{J}_{vv}^{*\mathrm{T}}\boldsymbol{J}_{vv}^{*}\mathrm{d}u\mathrm{d}v$ can be obtained in the same way.

Therefore, all integrals, $\iint \boldsymbol{J}^{*T}\boldsymbol{J}^{*}\mathrm{d}u\mathrm{d}v$, $\iint \boldsymbol{J}^{*T}\,du\,dv$, $\iint \boldsymbol{J}_u^{*T}\boldsymbol{J}_u^{*}\,du\,dv$, $\iint \boldsymbol{J}_v^{*T}\boldsymbol{J}_v^{*}\mathrm{d}u\mathrm{d}v$, $\iint \varphi \boldsymbol{J}_{uu}^{*T}\boldsymbol{J}_{uu}^{*}\mathrm{d}u\mathrm{d}v$, $\iint \varphi \boldsymbol{J}_{uv}^{*T}\boldsymbol{J}_{uv}^{*}\mathrm{d}u\,dv$, and $\iint \varphi \boldsymbol{J}_{vv}^{*\mathrm{T}}\boldsymbol{J}_{vv}^{*}\mathrm{d}u\mathrm{d}v$, can be obtained. Note that these integrals only depend on the parameter N, which seldom exceeds 20 in application settings, so we can compute these integrals in advance with N varying from 3 to 20, and save them in a file to be loaded at the beginning of the application. In our experiments, MATLAB is used to fulfill the job. Hence, the dynamic parameters $\boldsymbol{M}$, $\boldsymbol{D}$, $\boldsymbol{K}$ and $\boldsymbol{F}_{\mathrm{p}}$ can be obtained easily.

3 Numerical techniques in solving the dynamics equation

3.1 Solving the dynamics equation using central differences

For the Lagrangian dynamics equation, which is a second order differential equation, we will replace all derivatives by their discretized approximations first and then use the iteration method to solve the resulting time-varying system. In the previous work, backward differences are used to approximate the derivatives of the time $t+\Delta t$ (see Refs. [2—5]):

$$\ddot{\boldsymbol{C}}_0(t+\Delta t) = \frac{\boldsymbol{C}_0(t+\Delta t) - 2\boldsymbol{C}_0(t) + \boldsymbol{C}_0(t-\Delta t)}{\Delta t^2},$$

$$\dot{\boldsymbol{C}}_0(t+\Delta t) = \frac{\boldsymbol{C}_0(t+\Delta t) - \boldsymbol{C}_0(t-\Delta t)}{2\Delta t},$$

and then $\boldsymbol{C}_0(t+\Delta t)$ is obtained by solving the resulting linear equation. However, it is well known that the central difference is the best approximation to a derivative, so we will use central differences to discretize the dynamics equation. In detail,

$$\ddot{\boldsymbol{C}}_0(t) = \frac{\boldsymbol{C}_0(t+\Delta t) - 2\boldsymbol{C}_0(t) + \boldsymbol{C}_0(t-\Delta t)}{\Delta t^2},$$

$$\dot{\boldsymbol{C}}_0(t) = \frac{\boldsymbol{C}_0(t+\Delta t) - \boldsymbol{C}_0(t-\Delta t)}{2\Delta t},$$

and accordingly the dynamics equation $\boldsymbol{M}\ddot{\boldsymbol{C}}_0(t) +$

$\boldsymbol{D}\dot{\boldsymbol{C}}_0(t)+\boldsymbol{K}\boldsymbol{C}_0(t)=\boldsymbol{F}_s+\boldsymbol{F}_e(t)$ can be rewritten as

$$\begin{aligned}(2\boldsymbol{M}+\boldsymbol{D}\Delta t)\boldsymbol{C}_0(t+\Delta t)&=(4\boldsymbol{M}-2\Delta t^2\boldsymbol{K})\boldsymbol{C}_0(t)\\&+(\boldsymbol{D}\Delta t-2\boldsymbol{M})\boldsymbol{C}_0(t-\Delta t)\\&+2\Delta t^2\boldsymbol{F}_s+2\Delta t^2\boldsymbol{F}_e(t),\end{aligned}\tag{3}$$

where $\boldsymbol{F}_s=\boldsymbol{K}\boldsymbol{C}_0(0)$. Initially we assume that $\boldsymbol{C}_0(-\Delta t)=\boldsymbol{C}_0(0)$, and then we can solve the time-varying system (3) by recursively computing $\boldsymbol{C}_0((n+1)\Delta t)$ from the linear equation

$$\begin{aligned}(2\boldsymbol{M}+\boldsymbol{D}\Delta t)\boldsymbol{C}_0((n+1)\Delta t)&=(4\boldsymbol{M}-2\Delta t^2\boldsymbol{K})\boldsymbol{C}_0(n\Delta t)\\&+(\boldsymbol{D}\Delta t-2\boldsymbol{M})\boldsymbol{C}_0((n-1)\Delta t)+2\Delta t^2\boldsymbol{F}_s\\&+2\Delta t^2\boldsymbol{F}_e(n\Delta t),\end{aligned}\tag{4}$$

according to the known values $\boldsymbol{C}_0(n\Delta t)$ and $\boldsymbol{C}_0((n-1)\Delta t)$, until $\boldsymbol{C}_0((n+1)\Delta t)=\boldsymbol{C}_0(n\Delta t)$ approximately or the iterative time has reached the maximal value. Note that the coefficient matrix of the linear equation (4), i.e. $(2\boldsymbol{M}+\boldsymbol{D}\Delta t)$, is constant in the iterative procedure, so we can solve Eq. (4) very efficiently with only one time of LU-decomposition of the coefficient matrix.

3.2 Local solution for the linear system

It is obvious that all the m vertices are introduced in the linear equation (4), that is, the linear equation has an order of m. For a model containing a huge quantity of vertices and faces, we accordingly have to deal with a high-order linear equation with very high computational cost. On the other hand, the experiments tell us that one force influences a local region of the surface visibly but changes the other part slightly when the deformation of the surface is not too much. In many cases, only a few forces are applied on a complicated surface and the change of most part of the surface can be ignored. Hence we can solve the linear system approximately by fixing the vertices far from the action spots of the forces and reducing the linear equation (4) into a low-order equation.

We regard all vertices in the ρ rings around the vertex $\boldsymbol{v}$ as the influence region of a force applied to $\boldsymbol{v}$, denoted by $\mathrm{Reg}(\boldsymbol{v})$, where ρ is an integer. That is, $\mathrm{Reg}(\boldsymbol{v})$ is defined as

$$\mathrm{Reg}(\boldsymbol{v})=\bigcup_{k=0}^{\rho}\mathrm{ring}(\boldsymbol{v},k),$$

where $\mathrm{ring}(\boldsymbol{v},0)=\{\boldsymbol{v}\}$, and $\mathrm{ring}(\boldsymbol{v},k+1)$ denotes all vertices sharing a face with some vertex in $\mathrm{ring}(\boldsymbol{v},k)$ but not belonging to $\bigcup_{j=0}^{k}\mathrm{ring}(\boldsymbol{v},j)$, for any k. If all the n forces are applied to the vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, …, and $\boldsymbol{v}_n$ in the control mesh, respectively, then the total influence region is $\bigcup_{j=1}^{n}\mathrm{Reg}(\boldsymbol{v}_j)$, which is supposed to be $\{\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_{m_0}\}$. Then, we only solve these m_0 vertices from the linear system, while fixing all other vertices in the control mesh, that is, the linear equation (4) has been simplified from m-order to m_0-order.

Suppose that $\boldsymbol{V}_1$, $\boldsymbol{V}_2$, …, and $\boldsymbol{V}_{m_0}$ are the first m_0 vertices in $\boldsymbol{C}_0$ without loss of generality, then we divide $\boldsymbol{C}_0$ into two blocks:

$$\boldsymbol{C}_0=\begin{pmatrix}\boldsymbol{C}_0^1\\\boldsymbol{C}_0^2\end{pmatrix},$$

where $\boldsymbol{C}_0^1=(\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_{m_0})^{\mathrm{T}}$, and $\boldsymbol{C}_0^2$ is fixed. $\boldsymbol{M}$, $\boldsymbol{D}$, $\boldsymbol{K}$, $\boldsymbol{F}_s$ and $\boldsymbol{F}_e$ are divided into several blocks accordingly. For example,

$$\boldsymbol{K}=\begin{pmatrix}\boldsymbol{K}_1&\boldsymbol{K}_2\\\boldsymbol{K}_3&\boldsymbol{K}_4\end{pmatrix},$$

where $\boldsymbol{K}_1$ is an $m_0\times m_0$ matrix. Thus, the linear equation is simplified as

$$\begin{aligned}(2\boldsymbol{M}_1+\boldsymbol{D}_1\Delta t)\boldsymbol{C}_0^1((n+1)\Delta t)&=(4\boldsymbol{M}_1-2\Delta t^2\boldsymbol{K}_1)\boldsymbol{C}_0^1(n\Delta t)\\&+(\boldsymbol{D}_1\Delta t-2\boldsymbol{M}_1)\boldsymbol{C}_0^1((n-1)\Delta t)\\&+2\Delta t^2(\boldsymbol{F}_s+\boldsymbol{F}_e^1(n\Delta t)),\end{aligned}\tag{5}$$

where

$$\boldsymbol{F}_s=\boldsymbol{F}_s^1-\boldsymbol{K}_2\boldsymbol{C}_0^2=\boldsymbol{K}_1\boldsymbol{C}_0^1(0).$$

In other words, in order to solve the linear system (4) approximately, or find the local solution for system (4), we make $\boldsymbol{C}_0^2$ fixed and solve $\boldsymbol{C}_0^1$ from the linear system (5) by the iteration method.

4 Experiments and results

Let us glance at a control mesh obtained by subdividing a cube twice, where the set of control vertices of the original cube is $\{(x,y,z)^{\mathrm{T}}\,|\,|x|=|y|=|z|=1\}$. We apply four forces on the mesh as shown in Fig. 4(a), and solve the dynamics equation (1) using backward differences and central differences, respectively. It is shown in the experiment that the difference of the two resulting control meshes is 0.006443, which is very small. In fact, we cannot distinguish the two limit surfaces intuitively (see Fig. 4). However, the central differ-

ence is still a better choice at a mathematical angle, i.e. more approximate to the derivative. Certainly, the deviation of the dynamic models based on two types of differences grows when the varying rate of $\boldsymbol{F}_e(t)$ increases, then the advantage of central differences can be clearly seen.

To compare the global method and the local method for solving the time-varying system, an experiment is performed with an object like a hill, as shown in Fig. 5(a). Figure 5(b) shows the deformed surface obtained by solving the global linear equation (4), and Fig. 5(c) gives the resulting surface obtained by solving the local equation (5) and fixing all other vertices, where the influence region is painted with a dark color. It can be seen that the two surfaces solved by the linear systems (4) and (5) are intuitively uniform. In detail, the maximum distance between the vertices of the control mesh deformed by system (4) and the corresponding vertices computed from the system (5) is 0.003906. That is to say, the difference between the global solution and the local solution for the dynamics equation is very small and usually can be ignored. Moreover, the deformation time is 13.422s for the global method ($m=712$) in contrast to 0.296s for the local method ($m_0=81$). It is obvious that the local method greatly reduces the computing time and storage, especially for complicated models with a huge quantity of vertices and faces. It should be pointed out that open control meshes can also be introduced in the proposed framework without modifying the deformable model if the influence regions of external forces lie in the interiors of the meshes. In addition, ρ is set to be 4 in the experiments, which is large enough in general.

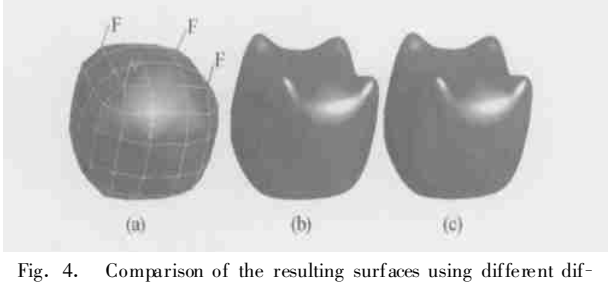

Fig. 4. Comparison of the resulting surfaces using different differences. (a) Initial control mesh; (b) using backward differences; (c) using central differences.

Using the modeling technique proposed above, we can modify the shapes of Doo-Sabin surfaces very freely. Some examples of deformable surfaces modeling are given in Fig. 6, where Fig. 6(a) and (b) are

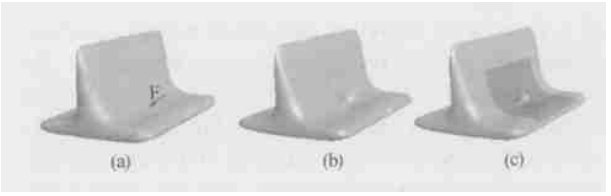

Fig. 5. Comparison of the global and local solutions. (a) The initial control mesh with one force; (b) the global solution of the dynamics equation; (c) the local solution with the influence region marked by a dark color.

the control polyhedron and the corresponding limit surface of Doo-Sabin subdivision, respectively, and Fig. 6(c)—(f) are the deformed surfaces via application of different external forces. For another example, the forces of a more complicated distribution are applied on a cup surface, which is designed with Doo-Sabin subdivision, and the original surface and the deformed shape are shown in Fig. 7, respectively.

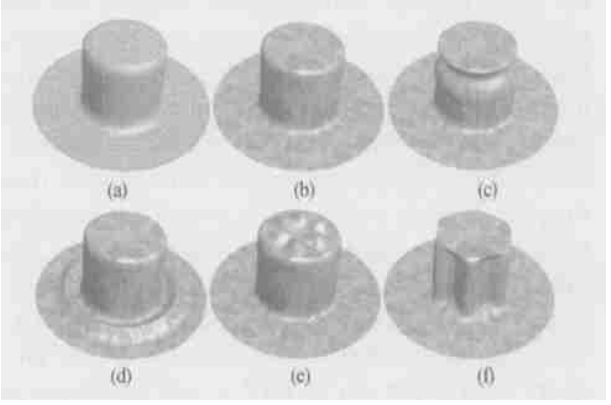

Fig. 6. Examples of hat modeling. (a) The control polyhedron; (b) the corresponding Doo-Sabin surface; (c)—(f) deformed surfaces via the applications of different forces.

Fig. 7. Cup modeling. (a) The cup surface based on Doo-Sabin subdivision; (b) the deformed surface by applying external forces.

5 Conclusions

Deformable subdivision surfaces have many applications in computer graphics, geometric modeling, medical imaging and scientific visualization. It is an intuitive and powerful tool for editing the shapes of subdivision surfaces. In this paper, we have present-

ed a deformable model of Doo-Sabin surfaces and extended the physics-based subdivision surfaces to the dual setting. In the model, the dynamic parameters are computed in a unified way for both non-defective and defective subdivision matrices, without subdividing the control mesh recursively, and the central differences are used to discretize the Lagrangian dynamics equation instead of the backward differences. After analyzing the dynamics system, we decompose the forces applied on the surface into two parts, so the physical background behind the deformable model becomes clearer. Further, we give a local scheme to solve the linear equation approximately, and the order of the linear equation is greatly reduced. Hence the computation time is shortened, and the needed storage is also reduced. It is shown in our experiments that the resulting surface obtained by the local scheme is almost uniform to the resulting surface obtained by the global scheme. Using the local scheme, one can easily introduce open control meshes into the proposed framework without modifying the deformable model if no boundary vertices lie in the influence regions of external forces. Moreover, our deformable dual subdivision model is more convenient to be handled than the prior dynamic models in the interactive modeling process with interlaced steps of applying forces and subdividing meshes. Therefore, our deformable model is more efficient than the existing dynamic models. The experiments demonstrate that the model proposed in the paper can be used to edit the shapes of subdivision surfaces very efficiently and flexibly. Other dual subdivision surfaces can be similarly applied in the proposed framework of deformable surfaces modeling.

References

1 Zorin D. and Schroder P. Subdivision for Modeling and Animation. In: SIGGRAPH' 2000 Course Notes, Course 23, New Orleans, Louisana, USA, 2000.

2 Qin H., Mandal C. and Vemuri B. C. Dynamic Catmull-Clark subdivision surfaces. IEEE Transactions on Visualization and Computer Graphics, 1998, 4(3): 215—229.

3 Mandal C., Qin H. and Vemuri B. C. A novel FEM-based dynamic framework for subdivision Surfaces. Computer-Aided Design, 2000, 32: 479—497.

4 Qin K., Wang H., Li D. et al. Physics-based subdivision surface modeling for medical imaging and simulation. In: Proceedings of MIAR' 2001, IEEE Computer Society Press, Hong Kong, 2001, 117—124.

5 Qin K., Chang Z., Wang H. et al. Physics-based Loop surface modeling. Journal of Computer Science and Technology, 2002, 17(6): 851—858.

6 Stam J. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In: Proceedings of SIGGRAPH' 98, Orlando, Florida, USA, 1998, 395—404.

7 Stam J. Evaluation of Loop subdivision surfaces. Appearing as an appendix in the CD disc of SIGGRAPH' 98, 1998.

8 Doo D. and Sabin M. Behavior of recursive division surfaces near extraordinary points. Computer-Aided Design, 1978, 10: 356—360.

9 Wang H. and Qin K. Precise evaluation of uniform Doo-Sabin surfaces. Progress in Natural Science, 2003, 13(5): 391—396.

10 Halstead M., Kass M. and DeRose T. Efficient, fair interpolation using Catmull-Clark surfaces. In: Proceedings of SIGGRAPH' 93, Anaheim, California, USA, 1993, 35—44.