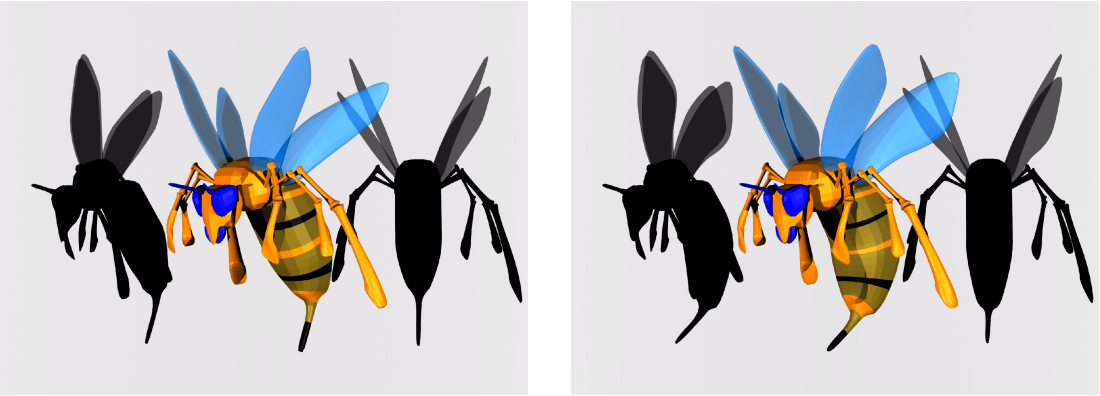
Fine Tuning: Curve and Surface Deformation by Scaling Derivatives

Kenjiro T. Miura
Shizuoka University
Department of Mechanical Engineering
Hamamatsu, Shizuoka, Japan
ktmiura@eng.shizuoka.ac.jp
Tel & Fax: 053-478-1074

Fuhua (Frank) Cheng
University of Kentucky, USA

Lazhu Wang
Raindrop Geomagic, Inc., USA



(a) Original surface (Doo-Sabin subdivision surface).

(b) Deformed surface (Non-uniform Catmull-Clark subdivision surface).

Figure 1. Example of subdivision surface fine tuning. The deformation process converts a Doo-Sabin subdivision surface to a non-uniform Catmull-Clark surface.

Abstract

A deformation-based fine tuning technique for parametric curves and surfaces is presented. A curve or surface is deformed by scaling its derivative, instead of manipulating its control points. Since only the norm of the derivative is adjusted, the resulting curve or surface keeps the basic shape of the original profile and curvature distribution. Therefore, the new technique is especially suitable for last minute fine tuning of the design process. Other advantages include: (1) the fine tuning process is a real local method, it can be performed on any portion of a curve or a surface, not just on a set of segments or patches; (2) by allowing a user to drag a scalar function to directly adjust the curvature (and, consequently, fairness) of a curve or surface, the new technique makes the shape design process more intuitive and effective; (3) the new technique is suitable for precise shaping and deforming such as making the curvature of a specific portion twice as big. In many cases, it can achieve results that other methods such as FFD can not; (4) the fine tuning process can also be used for subdivision curves and surfaces. Related techniques and test results are included.
Keywords: curve and surface deformation, subdivision curve and surface

1 Introduction

The generation of a smooth and fair curve or surface is an important and fundamental issue in computer graphics and computer aided geometric design. Smoothness is concerned with the functional and/or geometric continuity of a curve or surface (see, for example, [5]). Extensive research has been done in this area and various curve/surface generation methods have been proposed to ensure C^1 and C^2, or G^1 and G^2 continuity between adjacent curve segments or surface patches. Fairness is a more macroscopic concept than smoothness and its measure depends on fairness metrics. Many research works have been done to deal with fairness (cf.[14]). Nevertheless, it remains a difficult task to generate a fair curve or surface.

A major problem is the lack of fairness manipulation techniques. Curvature and variation of curvature are essential factors in determining the fairness of a curve or surface. Unfortunately, construction scheme of popular curve and surface representation schemes such as Bézier or B-spline, i.e., blending control points to generate curve or surface points, does not provide a user with direct manipulation capability of these quantities. Actually, since a simple relocation of the control points could change the curvature distribution dramatically, it is difficult to generate a fair curve or surface by directly manipulating their control points. The clothoid [6, 17] and QI [7] curves, on the other hand, have simple curvature profiles because the curves are defined by specifying their tangent vector directions. Hence, it is relatively easy to manipulate fairness of these curves. But, with limited shape range, these curves are not especially suitable for computer graphics applications.

A popular method in achieving fairness is to minimize a fairness functional of curvature or its variation to produce a fair free-form surface (Moreton and Séquin [8]). Such a method, however, tends to take large processing time and is not suitable for an interactive environment. Another problem with minimizing a fairness functional is the difficulty for a designer to perform local modification because of the global nature of the functional minimization process.

In the design process, it is desirable (sometimes necessary) to have a toolkit that, without changing the basic shape profile and curvature distribution of the object, allows the designer to slightly modify specific portions of the object to ensure a last minute fine tuning. One possible technique for this purpose is the so called free-form deformation (FFD) proposed by Sederberg and Parry [15]. FFD deforms an object by deforming a bounding volume defined by a 3D lattice of control vertices and a set of basis functions. This approach can easily control the shape profile and curvature distribution of the object by scaling the bounding volume as a whole, but it is not suitable for precise shaping and deforming, such as making the curvature of a specific portion of a curve twice as big. Other shortcomings of the FFD method include (1) the result may be unpredictable when the model is complex, and (2) it does not provide an analytic expression of the deformed model.

In this paper, we will address the above problems by proposing a deformation based *fine tuning* technique for parametric curves and surfaces. The deformation process is different from the previous approaches in that the deformation is performed by scaling the derivative of the curve or surface, instead of manipulating its control points, as shown in Figure 2. Since the scaling process changes the norm of the derivative only, the deformation fine tunes the curve or surface without changing the basic shape of its profile and curvature distribution. For instance, the red curve in Figure 2(a) is obtained by decreasing the norm of the green curve's derivative. Such a result can not be achieved by other deformation methods. The red curve in Figure 2(b) is similar to the red curve in (a) except it is also rotated and scaled to ensure the end points of the fine tuned curve remain the same as the original curve.

Other advantages of the new fine tuning method include: (1) the fine tuning process is a real local method, it can be performed on any portion of a curve or a surface, not just on a set of segments or patches; (2) by allowing a user to drag a scalar function to directly manipulate the curvature (and, consequently, fairness) of a curve or surface, the new technique makes the shape design process more intuitive and effective; (3) the new technique is suitable for precise shaping and deforming such as making the curvature of a specific portion twice as big. In many cases, it can achieve results that would not be possible using methods such as FFD; (4) the fine tuning techniques can also be used for subdivision surfaces as shown in Figure 1.

The remainder of this paper is organized as follows. The basics of the fine tuning technique for parametric curves is presented in Section 2. Fine tuning techniques for parametric surfaces are given in Section 3. The process of using the fine tuning technique on subdivision curves and surfaces is shown in Section 4. Concluding remarks and future research directions are given in Section 5.

(a) Original curve (green) and fine-tuned curve (red, the start point is fixed).

(b) Original curve (green) and fine-tuned curve (red, both end points are fixed).

Figure 2. Examples of curve fine tuning. The green curves are deformed by scaling their derivatives gradually to get the red curves.

2 Curve Fine Tuning

Basic ideas of the fine tuning technique for curves will be presented in this section. The technique can be used for all parametric curves. However, for simplicity of presentation, we will consider B-spline curves in this section only.

The idea of curve fine tuning is motivated by the following observation: given a curve $\boldsymbol{C}(t)$ with $0 \leq t \leq 1$, if one multiply the derivative $d\boldsymbol{C}(t)/dt$ by a scalar function $\alpha(t)$ and then integrate the product, one gets a curve $\boldsymbol{D}(t)$ with similar shape if the scalar function is close to 1. $\boldsymbol{D}(t)$ is defined as follows:

$$\begin{aligned} \boldsymbol{D}(t) &= \boldsymbol{C}(0) + \int_0^t \alpha(t) \frac{d\boldsymbol{C}(t)}{dt} dt \\ &= \boldsymbol{P}_0 + \int_0^t \boldsymbol{E}(t) dt. \end{aligned} \tag{1}$$

If $\boldsymbol{C}(t)$ and $\alpha(t)$ are both in the B-spline form, then the product $\alpha(t)d\boldsymbol{C}(t)/dt$ and $\boldsymbol{D}(t)$ are also B-spline curves.

The above equation also represents a deformation process of the curve driven by manipulating the derivative and, consequently, the curvature of the curve. For instance, if $\alpha(t)$ is set to 2 near a given parameter t_0, then the derivative of the curve at $t = t_0$ is scaled up by two and the curvature is reduced by one half. More generally, the curvature is changed by $1/\alpha(t)$ if the derivative is scaled by $\alpha(t)$. This claim is proved as follows: the curvature $\kappa(t)$ of $\boldsymbol{C}(t)$ is given by:

$$\kappa(t) = \frac{|\frac{d\boldsymbol{C}(t)}{dt} \times \frac{d^2\boldsymbol{C}(t)}{dt^2}|}{|\frac{d\boldsymbol{C}(t)}{dt}|^3}. \tag{2}$$

The first and second derivatives of $\boldsymbol{D}(t)$ are

$$\begin{aligned} \frac{d\boldsymbol{D}(t)}{dt} &= \alpha(t) \frac{d\boldsymbol{C}(t)}{dt}, \\ \frac{d^2\boldsymbol{D}(t)}{dt^2} &= \frac{d\alpha(t)}{dt} \frac{d\boldsymbol{C}(t)}{dt} + \alpha(t) \frac{d^2\boldsymbol{C}(t)}{dt^2}. \end{aligned} \tag{3}$$

Then the curvature $\kappa_D(t)$ of $\boldsymbol{D}(t)$ is given by:

$$\kappa_D(t) = \frac{|\alpha(t)\frac{d\boldsymbol{C}(t)}{dt} \times (\frac{d\alpha(t)}{dt}\frac{d\boldsymbol{C}(t)}{dt} + \alpha(t)\frac{d^2\boldsymbol{C}(t)}{dt^2})|}{|\alpha(t)\frac{d\boldsymbol{C}(t)}{dt}|^3}. \tag{4}$$

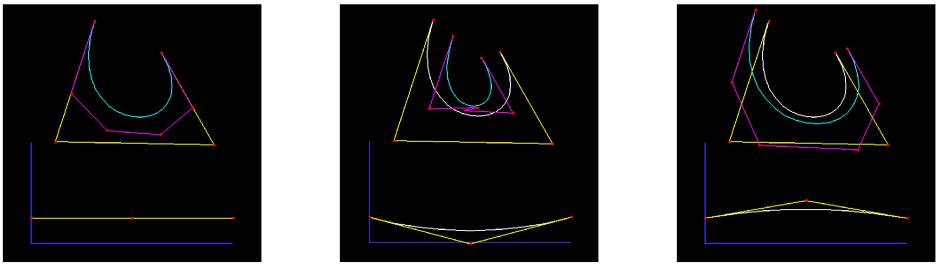
Since the cross product of two vectors oriented in the same direction is 0,

$$\begin{aligned} \kappa_D(t) &= \frac{1}{\alpha(t)} \frac{|\frac{d\boldsymbol{C}(t)}{dt} \times \frac{d^2\boldsymbol{C}(t)}{dt^2}|}{|\frac{d\boldsymbol{C}(t)}{dt}|^3} \\ &= \frac{\kappa(t)}{\alpha(t)}. \end{aligned} \tag{5}$$

The above equation proves the claim.

We call this type of shape manipulation technique *shape fine tuning* because, like fine tuning the pitch of a violin, shape manipulation through derivative scaling would gradually scale the curve and only induce minor change on the curvature distribution of the curve if the value of the scalar function does not change dramatically. This fine tuning process usually does not change the number of local extremes and would only change their locations slightly if the value of the scalar function does not exceed certain range.[1] Figure 3 shows several examples of the curve deformed by the fine tuning technique. If the scalar function is an identity function, the original and fine-tuned curves are identical as shown in Fig. 3(a). Figure 4 shows the curvature profile of these curves. The curvature of the fine-tuned curves are gradually scaled according to the scale function values as claimed above. Although the positions of the control points of the fine-tuned curve in Fig. 3(b) are strongly zigzaged, the curvature of the curve is only gradually scaled and it does not exhibit occurence of undulations.

A positive scalar function is usually used in the fine tuning process to avoid having a tangent in the opposite direction. For practical use of the fine tuning technique, a scalar function of degree $n-1$ is recommended for a curve of degree n. This is because a curve of degree n is C^{n-1} continuous. To maintain the same degree of continuity for the fine tuned curve, the scalar function $\alpha(t)$ should be at least C^{n-2} continuous, i.e. of degree $n-1$, as one can see from Eq.(1). Therefore, if the original curve is cubic, $\alpha(t)$ should be quadratic, and the fine-tuned curve is quintic. If the original curve is quadratic, $\alpha(t)$ should be linear and the fine-tuned curve is cubic. We use these combinations of degrees for all the examples in this paper, including the surface case.



(a) Original and fine-tuned curves are identical.

(b) Original (white) and fine-tuned (cyan) curves.

(c) Original (white) and fine-tuned (cyan) curves.

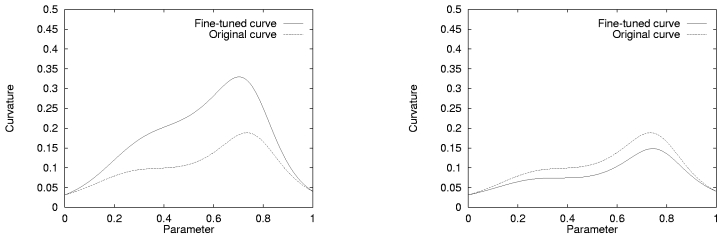
Figure 3. Examples of curve fine tuning. The white curves are deformed by scaling their derivatives gradually to get the cyan curves.

Implementation of the fine tuning technique for B-spline curves is straightforward except for the calculation of the product of a B-spline curve and a B-spline scalar function. The production can be regarded as a degree elevation process and implementation details can be found in standard textbooks on B-spline curves (see, for instance, [11]). A complete discussion on the product of two NURBS functions can be found in Morken [9]. Implementation of the fine tuning technique for other parametric curves is similar except clothoid curves whose derivatives can not be integrated analytically.

2.1 Boundary Constraints for Curves

The fine tuning technique preserves tangent direction of the curve at the endpoints. But it does not preserve the endpoints of the curve if the scalar function is not equal to 1. It is often necessary to keep the positions of the end points of the curve, especially to deform it locally. One simple method to satisfy the end-point constraint without changing the basic shape of the curvature profile is to apply a rotation and a scaling after the deformation

[1] This statement is rather qualitative. If $\alpha(t)$ is overly modified, some extreme values would vanish and new ones would be generated.

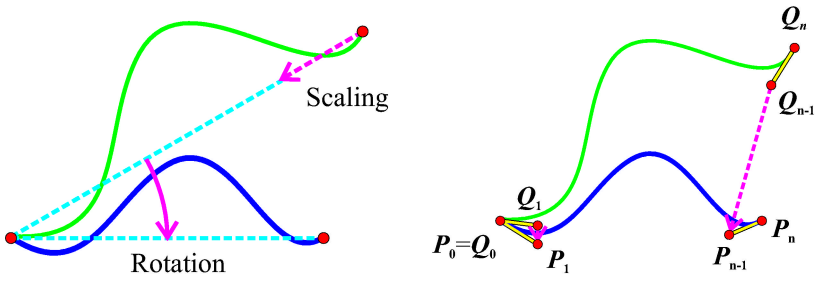


(a) Original curve (dashed) and fine-tuned curve (solid).

(b) Original curve (dashed) and fine-tuned curve (solid).

Figure 4. Curvature profiles.

process, as shown in Figure 5(a). The red curves in Figures 2(a) and 2(b) are obtained by fine tuning the given curve in green. The curve in (a) is deformed without constraint on the endpoints. The curve in (b) is deformed by the same scalar function, but also rotated and scaled to satisfy the end-point constraint. Hence the red curves in (a) and (b) are geometrically similar. Curvature profile is invariant under rotation and is only globally scaled by global scaling of the curve. Therefore none of these geometric transformations would cause undesirable undulations to the curvature profile.



(a) End-point constraint.

(b) Tangent constraint.

Figure 5. Rotation and Scaling. (a) End-point constraint. One simple method to hold the endpoints unchanged is to apply a rotation and a scaling after the fine tuning process. (b) Tangent constraint. The second control points Q_1 and Q_{n-1} from both endpoints Q_0 and Q_n of the fine-tuned curve are translated, rotated and scaled to match the corresponding control point P_1 and P_{n-1} of the invariant curve.

In addition to having fixed endpoints, sometimes it is also necessary to have fixed tangent and curvature at the endpoints. These constraints can be satisfied in a similar way as the end-point constraint. For the tangent constraint, first, multiply the given curve's derivative by an identity scalar function $\beta(t)$ of the same order and knot vector as $\alpha(t)$ and compute the control points of the fine-tuned curve (the blue curve in Figure 5(b)) which is the same as the original curve (but of higher degree). Then, apply $\alpha(t)$ to the original curve, and rotate and scale the fine tuned curve (the green curve in Figure 5(b)) so that the second control points on both ends also match the second control points on both ends of the above (blue) curve.

The curvature constraint is satisfied by matching both the second and third control points on both ends. Figure 6 shows examples of the end-point, tangent, and curvature constraints. Note that the more the constraints, the less freedom the deformation process.

The above method for the end-point constraint does not work for closed curves because an adequate scaling factor for the scaling process can not be determined for a closed curve. A different approach, based on a physical model described in Section 4 for recursive subdivision curves, has to be used.

2.2 Local Fine Tuning of a Curve

With the capability of fixing the endpoints, the tangents and the curvature of a curve segment at the endpoints, one can actually perform fine tuning on any portion of a parametric curve. For instance, to fine tune a portion

whose endpoints are A and B of a B-spline curve $\boldsymbol{C}(t)$, simply apply the above fine tuning process to the segment between A and B with the fixed endpoints, fixed endpoint tangents and fixed endpoint curvature constraints.

Examples of local fine tuning and their FFD counterparts are shown in Figure 6 and their curvature profiles are drawn in Fingure 6. 6(a) and 6(b) are deformation results by FFD of a bicubic B-spline surface whose grid is shown in light red. 6(c) and 6(d) are results of the fine tuning process of the same cubic B-spline curve (in white). The scalar functions used in these cases, shown at the bottom of the figures, are quadratic B-spline functions. The end-point constraint, as discussed in the previous section, is enforced in both cases.

The results in (a) and (b) show that FFD is clearly better than direct manipulation of the control points in the sense that moving a point of the grid provides smoother shape deformation of the curve than moving a control point. However, the impact of moving a grid point on the curvature profile of a curve is not so clear. Another problem is the selection of the grid size which determines the range of deformation in the FFD process.

The fine tuning technique, on the other hand, does not have these kinds of problems. The region to be fine tuned can be explicitly specified by the user and by dragging a scalar function, the user can adjust the derivative and, consequently, curvature profile of the curve systematically. The deformation process is completely local and unnecessary changes to the curvature profile can be reduced to a minimum, as shown in (c) and (d). Oscillations of the curvature profile could occur when imposing the tangent and curvature constraints at the endpoints of the deformed region. This is because increased constraints reduce the degree of freedom in the fine tuning process. But the oscillations are usually limited to small regions near the endpoints only, as shown in 6(e) and 6(f).

2.3 Repeated Local Fine Tuning

Figures 8(a) and 8(b) show the results of two consecutive local deformations by the fine tuning technique. In 8(a), the left eye of the given quadratic curve (in green) is locally deformed with the tangent boundary constraint to get a new shape (in red). The scalar function used in the deformation process is a linear function. Hence, the new shape of the left eye (in red) is a cubic curve. A degree reduction process [11] is then performed on this new curve to reduce its degree to two again. This new curve, together with a neighboring region, is then locally deformed with the tangent boundary constraint to get the red curve in 8(b).

This example demonstrates that, by performing an automatic degree reduction after each deformation process, one can locally fine tune a curve repeatedly without increasing the degree of the curve. Several techniques are available to evaluate the error induced by the degree reduction process [10]. One can control the number of segments in a fine-tuned curve by inserting a new knot into the curve only when the error of the degree reduction process for a segment exceeds a given tolerance.

2.4 Ornamental Curves

Figure 8(c) shows an example that can not be produced by FFD or simple global scaling. The doubly looped green curve is deformed by gradually decreasing the norm of its tangent for the first half and then gradually increasing the norm of its tangent for the second half to yield the red curve. As the scaling factor at every portion of the curve is strictly controlled by the scalar function, we can generate a geometrically regulated curve that may be used for ornamental purpose.

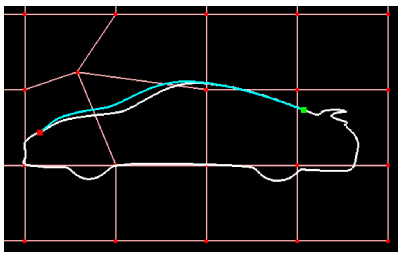
3 Surface Fine Tuning

Basic ideas of the surface fine tuning technique will be discussed in this section. For simplicity, only B-spline surfaces will be considered in this section. But, like the curve case, the technique can be used for all parametric surfaces except a few cases where analytic integration is not possible.

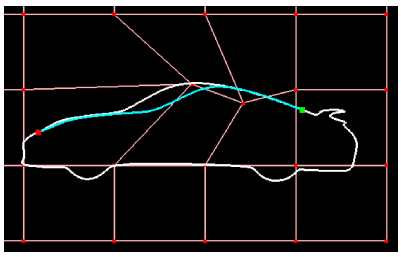
3.1 Curve Based Approach

Skinning is a basic but popular design method. A smooth surface is generated by interpolating (blending) a set of cross-sectional curves with some functional or geometrical continuity constraints. The shape of the resulting surface can be deformed by fine tuning its cross-sectional curves. An example is shown in Figure 6 where three cross-sectional curves drawn in green in (a) are locally fine tuned separately, as shown in (b), to generate the new surface shown in (c). The cross-sectional curves are interpolated with G^1 continuity.

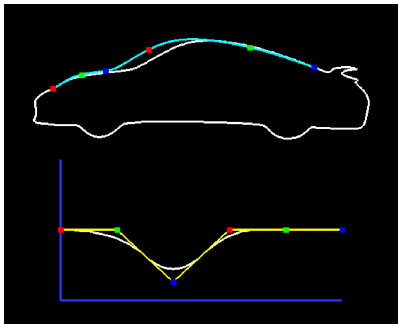
In the skinning process, the cross-sectional curves are usually segmented at the same longitudinal locations to ensure fairness of the resulting surface. In Figure 9, to ensure the same segmentation of the cross-sectional curves



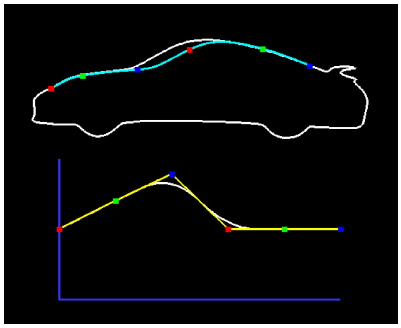
(a) Deformation by FFD (example No.1).



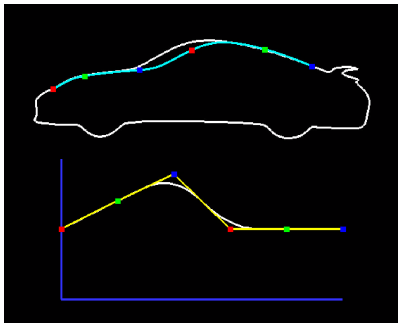
(b) Deformation by FFD (example No.2).



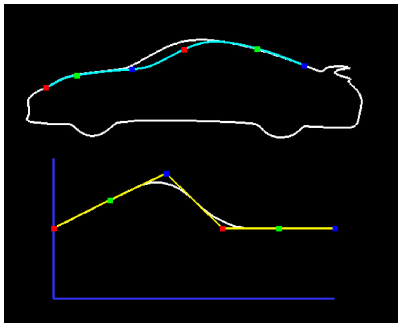
(c) Deformation by Fine Tuning (example No.1.).



(d) Deformation by Fine Tuning (example No.2.).

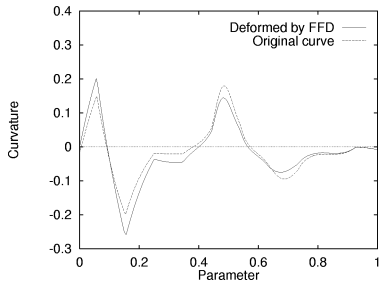


(e) Tangent constraint.

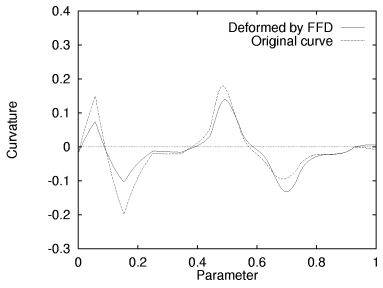


(f) Curvature constraint.

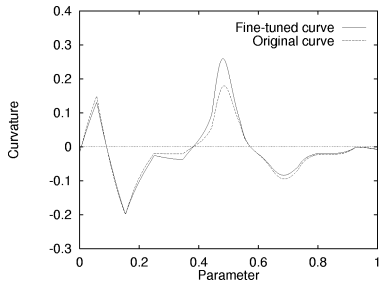
Figure 6. Comparison of fine tuning and FFD. The original curves and deformed curves are in white and cyan, respectively. The endpoints of the deformed curves are fixed. (a) and (b) are deformed by FFD. (c) and (d) are deformed by fine tuning. In (c) and (d), corresponding points of the fine-tuned curve and the control values of the scalar function are displayed in the same colors to show the relationship between the fine-tuned curve and the scalar function. (e) Tangent constraint. (f) Curvature constraint.



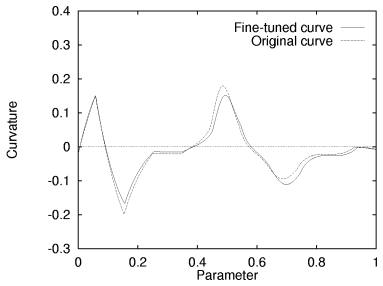
(a) Fig.6(a).



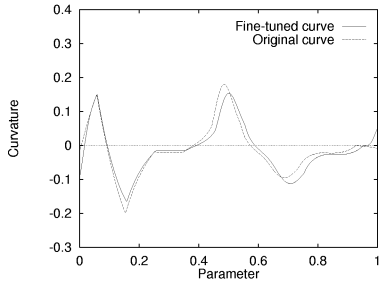
(b) Fig.6(b).



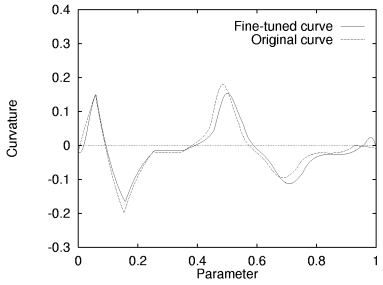
(c) Fig.6(c).



(d) Fig.6(d).



(e) Fig.6(e).

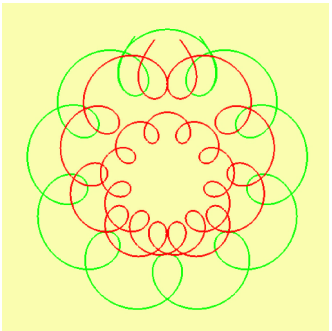


(f) Fig.6(f).

Figure 7. Curvature profiles.

(a) 1st local deformation with the tangent constraint.

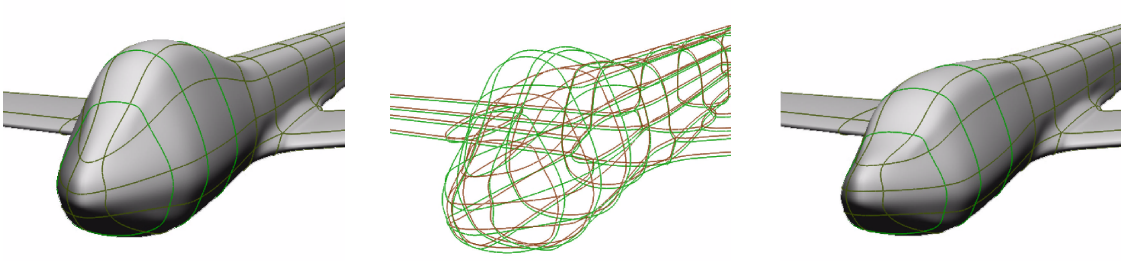
(b) 2nd local deformation with the tangent constraint at different endpoints.



(c) Ornamental curve.

Figure 8. Examples of curve fine tuning. (a) and (b): Repeated local deformation. (c) Ornamental curve.

is kept after the fine tuning process, a scalar function that has the same segmentation as the cross-sectional curves is used in the fine tuning process. A degree reduction process is then performed on the fine-tuned curves to reduce their degrees to the original ones and, consequently, the original segmentation.



(a) Original surface. (b) Comparison of the original and fine-tuned surfaces. (c) Fine-tuned surface.

Figure 9. Surface deformation based on curve fine tuning. In (a), three sectional curves in green are deformed using curve fine tuning technique to get the deformed surface in (c).

3.2 Surface Based Approach

Given a B-spline surface $\boldsymbol{S}(u,v)$, $0 \le u,v \le 1$, similar to the curve case, one can control its curvature at any portion by introducing a scalar function $\alpha(u,v) > 0$ as follows:

$$\boldsymbol{T}(u,v) = \boldsymbol{P}_0 + \int_{path} \alpha(u,v)\Big(\frac{\partial \boldsymbol{S}(u,v)}{\partial u} + \frac{\partial \boldsymbol{S}(u,v)}{\partial v}\Big)dudv, \tag{6}$$

where $\boldsymbol{P}_0 = \boldsymbol{S}(0,0)$. Ideally, $\boldsymbol{T}(u,v)$ should be independent of the integration path, i.e., $\partial(\alpha(u,v)\partial \boldsymbol{S}(u,v)/\partial u)/\partial v = \partial(\alpha(u,v)\partial \boldsymbol{S}(u,v)/\partial v)/\partial u$. However, this condition puts a restriction on the variation of $\alpha(u,v)$ and it restricts the user's capability in deforming the surface freely. Hence, instead of Eq.(6), the following definition is adopted for $\boldsymbol{T}(u,v)$:

$$\begin{aligned}\boldsymbol{T}(u,v) &= \boldsymbol{P}_0 + \frac{1}{2}\Big(\int_0^u \alpha(u,0)\frac{\partial \boldsymbol{S}(u,0)}{\partial u}du + \int_0^v \alpha(u,v)\frac{\partial \boldsymbol{S}(u,v)}{\partial v}dv \\ &\qquad + \int_0^v \alpha(0,v)\frac{\partial \boldsymbol{S}(0,v)}{\partial v}dv + \int_0^u \alpha(u,v)\frac{\partial \boldsymbol{S}(u,v)}{\partial u}du\Big) \\ &= \frac{1}{2}(\boldsymbol{T}_{uv}(u,v) + \boldsymbol{T}_{vu}(u,v)),\end{aligned} \tag{7}$$

where

$$\begin{aligned}\boldsymbol{T}_{uv}(u,v) &= \boldsymbol{P}_0 + \int_0^u \alpha(u,0)\frac{\partial \boldsymbol{S}(u,0)}{\partial u}du + \int_0^v \alpha(u,v)\frac{\partial \boldsymbol{S}(u,v)}{\partial v}dv, \\ \boldsymbol{T}_{vu}(u,v) &= \boldsymbol{P}_0 + \int_0^v \alpha(0,v)\frac{\partial \boldsymbol{S}(0,v)}{\partial v}dv + \int_0^u \alpha(u,v)\frac{\partial \boldsymbol{S}(u,v)}{\partial u}du.\end{aligned} \tag{8}$$

Eq.(7) requires two paths to calculate a point $\boldsymbol{T}(u,v)$, therefore, is twice as time-consuming as the one-path approach using either $\boldsymbol{T}_{uv}(u,v)$ or $\boldsymbol{T}_{vu}(u,v)$ only. But the two-path approach provides more flexibility in manipulating the value of the scalar function while the one-path approach might lose fine tuning effect in one parameter direction in some cases. Consider, for example, the case shown in Figure 10 where the value of the scalar function is modified in the region of the parameter space (u,v) painted in blue. In this case, if the one-path approach is adopted, say the red path, one would not get any fine tuning effect in u direction at all.

If fine tuning effect is needed only in a specific direction, one should adopt an adequate one-path integration, instead of the two-path approach.

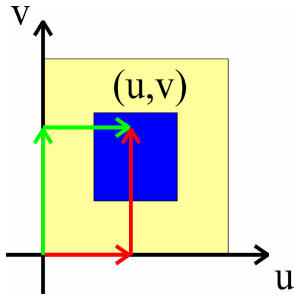
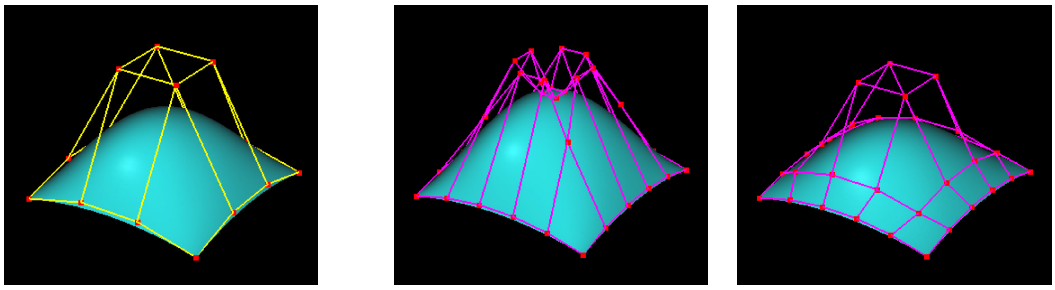


Figure 10. Advantage of the two-path approach.

3.3 Boundary Constraints for Surfaces

There are many occasions in the design process that one needs to hold the boundary curves of a surface unchanged while manipulating or deforming the surface. Fixed boundary implies that integration along the boundary curves is not required in calculating $\boldsymbol{T}(u, v)$, only the integrals along an internal path are needed.

Once the control points of the surfaces $\boldsymbol{T}_{uv}(u, v)$ and $\boldsymbol{T}_{vu}(u, v)$ defined by Eq.(8) have been computed, the boundary condition is achieved by applying the end-point constraint for the curve case to each row (column) of control points of $\boldsymbol{T}_{uv}(u, v)$ and $\boldsymbol{T}_{vu}(u, v)$ in v and u directions, respectively. Consider, for example, $\boldsymbol{T}_{uv}(u, v)$. Since the internal path is in v direction, each column of control points of $\boldsymbol{T}_{uv}(u, v)$ in v direction is relocated by a rotation and a scaling to enforce the end-point constraint of that column. The same operation is performed on rows of control points of $\boldsymbol{T}_{vu}(u, v)$ in u direction. Figure 11 shows examples of fine-tuned surfaces with the end-point(boundary) constraint. The original surface is a bi-cubic B-spline surafce and a bi-quadratic B-spline surface with 3×3 control values is used as the scalar function and the surface is fine tuned by decreasing and increasing only the center control value from 1 to -1.7 and to 16, respectively, using the two-path integration.



(a) Original surface.

(b) Fine-tuned (central control value $= -1.7$).

(c) Fine-tuned surface (central control value $= 16$).

Figure 11. Examples of surface fine tuning. (a) Original surface. (b): Fine-tuned surface (central control value $= -1.7$). (c) Fine-tuned surface (central control value $= 16$).

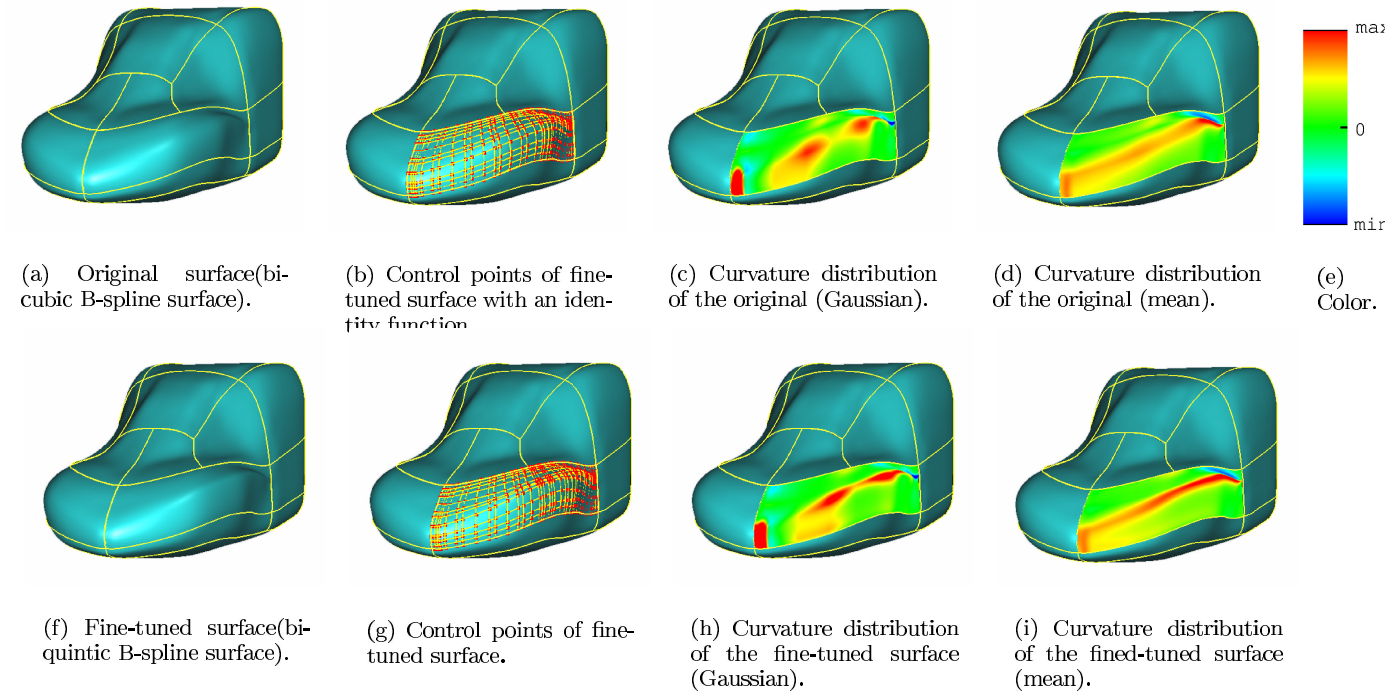
Tangent and curvature continuity across the boundary curves can be accomplished in the same manner as the curve case. For example, to maintain tangent continuity across the boundary curves, the second lows (columns) of the control points from the shared boundaries of the adjacent surfaces have to be fixed and matched with those of the fine-tuned surface. This is done by multiplying the derivative of the fine-tuned surface with an identity scalar function which is equal to 1 everywhere, and then by applying $\alpha(u, v)$ and by performing translation, rotation and scaling. For curvature, the third lows (columns) of the control points should be matched.

Because of the same reason as the curve case, the above method does not work for closed surfaces. Like the curve case, closed surfaces need different treatment which will be discussed in Section 4.

Figure 12 shows an example of fine-tuned surfaces with the curvature constraint. A bi-quadratic B-spline surface with 3×4 control values is used as the scalar function $\alpha(u, v)$. All the control values of the scalar function are set to 1 except the two central control ones which are set to -0.8 in this example.[2] Since the surface to be fine tuned

[2]The control value itself is negative, but the value of the scalar function is always positive and the direction of the derivatives are not reversed.

is much longer in one parameter direction, the one-path integration approach in the other (shorter) parameter direction is adopted for the fine tuning process. Since the small values are specified for the central control values, the middle part of the surface is sharpened along the longer parameter direction. The basic shape of curvature distribution is kept as shown in (c), (d), (h), and (i).



(a) Original surface(bi-cubic B-spline surface).

(b) Control points of fine-tuned surface with an identity function.

(c) Curvature distribution of the original (Gaussian).

(d) Curvature distribution of the original (mean).

(e) Color.

(f) Fine-tuned surface(bi-quintic B-spline surface).

(g) Control points of fine-tuned surface.

(h) Curvature distribution of the fine-tuned surface (Gaussian).

(i) Curvature distribution of the fined-tuned surface (mean).

Figure 12. Example of surface fine tuning. (a) Original surface. (b) The control points of the original surfaces. (c) and (d) are Gaussian and mean curvature distributions of the original. (e) Color assignment for curvature display. (f) Fine-tuned surface. (g) The control points of the fine-tuned surface. (h) and (i) are Gaussian and mean curvature distributions of the fined-tuned surface.

3.4 Local Fine Tuning of a Surface

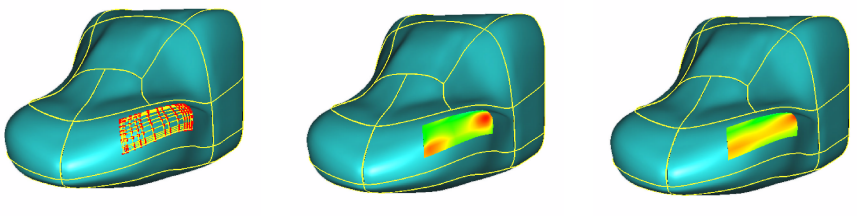
With the capability of holding the boundary curves unchanged during the fine tuning process and maintaining tangent and curvature continuity across the boundary curves of a surface patch, one can actually perform fine tuning on any rectangular portion of a parametric surface. An example is shown in Figure 13. A bi-quadratic B-spline surface with 3×3 control values is used as the scalar function and the surface is locally flatted by increasing only the center control value from 1 to 4 using the one-path integration.

3.5 Processing Time

Table 1 summarizes the processing times of the test cases shown in the paper. The platform is a Pentium III 1GHz computer.[3] For each test case, we show the degree and number of control points of the original shape, the degree and number of control values of the scalar function, and the processing time. The processing time covers all the operations performed, i.e., differentiation, product, and integration.

In the case of curve fine tuning, as can be seen from the table, the processing times of even rather complicated shapes are small enough for an interactive environment. The processing times of surface fine tuning, on the other hand, can not completely achieve interactive effect yet. Our current implementation is based on the recurrence relation of the B-splines [9]. A significant speedup can be expected by tabulating the product formulas with

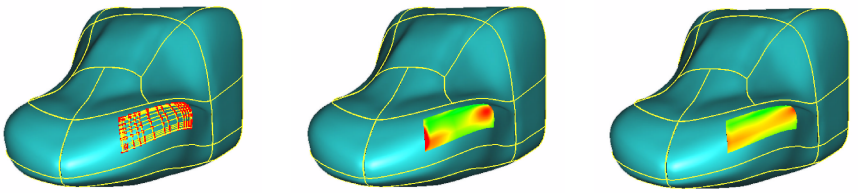
[3]All the processing times in the paper are measured on a Pentium III 1 GHz machine.



(a) Control points of the fine-tuned surface (boundary constraint).

(b) Gaussian curvature distribution (boundary constraint).

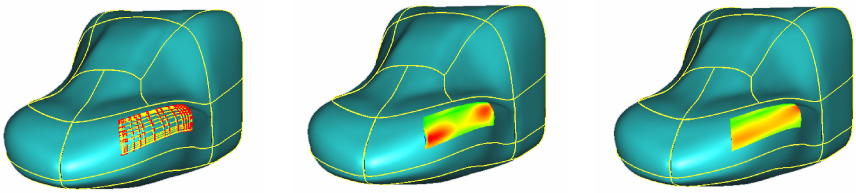
(c) Mean curvature distribution (boundary constraint).



(d) Control points of the fine-tuned surface (tangent constraint).

(e) Gaussian curvature distribution (tangent constraint).

(f) Mean curvature distribution (tangent constraint).



(g) Control points of the fine-tuned surface (curvature constraint).

(h) Gaussian curvature distribution (curvature constraint).

(i) Mean curvature distribution (curvature constraint).

Figure 13. Examples of local fine tuning. (a), (b) and (c) are the control points, Gaussian and mean curvature distribution of the fine-tuned surface with the boundary constraint. (d), (e) and (f), (g), (h) and (i) are those with the tangent and curvature constraints, respectively.

the symbolic operators for NURBS curves and surfaces proposed by Piegl and Tiller [12] for typical B-spline combinations.

Curve	deg. (orig.)	#ctl. pnt.	deg. (sca.)	#ctl. val.	time(second)
Face (Fig.2)	2	181	1	2	0.0052
Car (Fig.6)	3	72	2	6	0.013
Double-loop (Fig.8(c))	3	159	2	3	0.014
Surface	deg. u, v (orig.)	#ctl. pnt.	deg. u, v (sca.)	#ctl. val.	time(second)
Fig.11	3, 3	$4 \times 4 = 16$	2, 2	$3 \times 3 = 9$	0.16
	3, 3	$4 \times 4 = 16$	2, 2	$3 \times 4 = 12$	0.33
	3, 3	$5 \times 5 = 25$	2, 2	$3 \times 3 = 9$	0.49
Train (Fig.13)	3, 3	$4 \times 8 = 32$	2, 2	$3 \times 3 = 9$	0.61
Train (Fig.12)	3, 3	$6 \times 12 = 72$	2, 2	$3 \times 4 = 12$	2.74

Table 1. Table of degrees, numbers of control points of the original shape, degrees and numbers of control values, and processing times of fine tuning process (differentiation, product and integration).

4 Fine Tuning Subdivision Curves and Surfaces

Subdivision surfaces are powerful tools for graphical modeling and animation because of their scalability, numerical stability, simplicity in coding and, especially, their ability to represent complex shape of arbitrary topology [3]. A surface manipulation technique that does not work for subdivision surfaces would not be complete. In this section, we will show that the new fine tuning technique works for subdivision curves and surfaces as well. Three issues have to be addressed when dealing with closed subdivision curves and surfaces, namely, (1) how should the domain of the scalar function α be specified, (2) how should the product of the original shape and the scalar function be performed, and (3) how to keep the fine-tuned curve or surface closed. We will address these issues while we do the illustration.

4.1 Fine Tuning Subdivision Curves

We will use Chaikin's algorithm as an example to illustrate the fine tuning process of a subdivision curve. Chaikin's algorithm [2] generates a quadratic B-spline curve from a control polygon through recursive subdivision. Each subdivision step generates two new points on each polygon leg. If there are $n+1$ vertices $\boldsymbol{p}_i^j$, $i = 0, ..., n$, after the j-th recursive subdivision, then the two new points generated for the polygon leg $\boldsymbol{p}_i^j \boldsymbol{p}_{i+1}^j$ are defined as follows:

$$\boldsymbol{p}_{2i}^{j+1} := \frac{3}{4}\boldsymbol{p}_i^j + \frac{1}{4}\boldsymbol{p}_{i+1}^j, \tag{9}$$

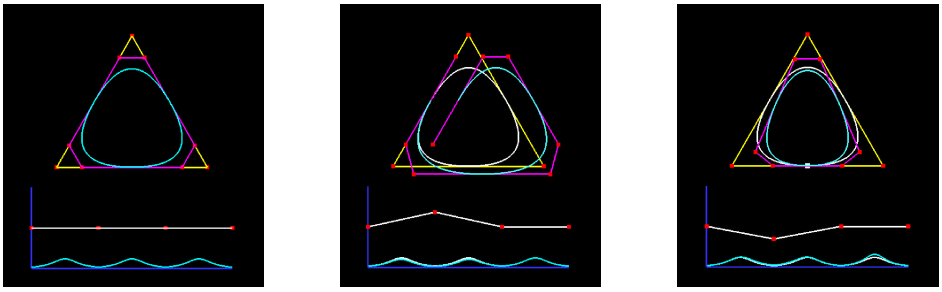
$$\boldsymbol{p}_{2i+1}^{j+1} := \frac{1}{4}\boldsymbol{p}_i^j + \frac{3}{4}\boldsymbol{p}_{i+1}^j. \tag{10}$$

The subdivision curve generated by Chaikin's algorithm is a quadratic B-spline curve. Hence, if a linear B-spline scalar function is used in the fine tuning process of the subdivision curve, the resulting curve would be a cubic B-spline curve. Consider, for example, the triangular control polygon drawn in yellow in Figure 14(a). The subdivision curve generated from this control polygon has three segments. If we apply as a scalar function a linear B-spline function of three segments whose knot vector is identical to that of the original curve but without the first and the last knots, we will obtain a three-segment, non-uniform cubic B-spline curve whose knots are all of multiplicity two (see Appendix (a)). Here the domain issue of the scalar function is resolved by adopting the same domain as the original curve, and the product issue is resolved by treating it as a degree elevation process which converts a uniform quadratic B-spline curve to a non-uniform cubic B-spline curve.

The conversion of the control polygon of a quadratic B-spline curve to that of a non-uniform cubic B-spline curve is accomplished by generating two new points defined as follows for each polygon leg $\boldsymbol{p}_i \boldsymbol{p}_{i+1}$,

$$\boldsymbol{q}_{2i} := \frac{5}{6}\boldsymbol{p}_i + \frac{1}{6}\boldsymbol{p}_{i+1}, \tag{11}$$

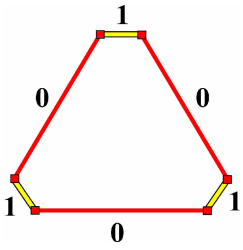
$$\boldsymbol{q}_{2i+1} := \frac{1}{6}\boldsymbol{p}_i + \frac{5}{6}\boldsymbol{p}_{i+1}, \tag{12}$$



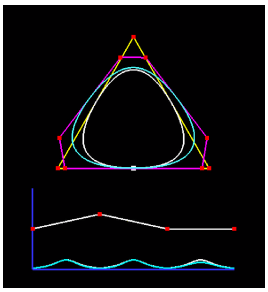
(a) Original control polygons (Quadratic: yellow; non-uniform cubic: magenta).

(b) Deformed subdivision curve (the start point is fixed).

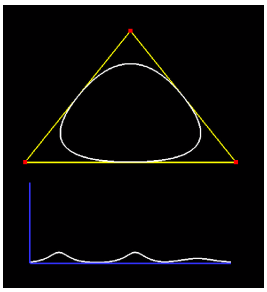
(c) Deformed curve based on the spring model.



(d) Knot intervals for the non-uniform cubic B-spline subdivision curve.



(e) Deformed curve using the same setting as (b) but based on the spring model.



(f) Deformed curve by changing the length of the bottom leg of the original curve.

Figure 14. Examples of subdivision curve fine tuning.

in a manner similar to Chaikin's algorithm. The converted control polygon is the control polygon of a non-uniform cubic B-spline subdivision curve proposed by Sederberg et al. [16]. For a periodic cubic B-spline curve, there is an one-to-one correspondence between the edges of the control polygon and the segments of the curve. Following the same notation of [16], a knot interval between two consecutive knots is assigned to each control edge of the converted curve, as shown in 14(d).

Note that for a periodic quadratic B-spline curve, there is an one-to-one correspondence between the vertices of the control polygon and the segments of the curve. However, for a periodic linear B-spline curve the correspondence is between the edges of the control polygon and the segments. To define a correspondence between the scalar function and the subdivision curve, we assume there is a virtual vertex between each two consecutive segments of the subdivision curve and assign a control value of the scalar function to the virtual vertex instead of the vertex of the original curve. Using the control values assigned at the ends of a quadratic segment of the original curve, one can define a linear function by interpolating these control values and use the value of this linear function to scale the derivative of corresponding point of the segment. This way, one can fine tune each segment of the curve and, consequently, the entire curve.

Figure 14(a) shows the original curve and its invariant version fine tuned by a linear identity scalar function. The graph beneath it is the scalar function with its control values (points) shown in red. The first and last control values are identical to ensure that the scalar function is also periodic. 14(b) shows fine-tuned curves generated by increasing the control value corresponding to the virtual vertex between the two segments joined at the bottom of the original curve. The curve is no longer closed after the fine tuning process. The method described in Section 2 for the end-point constraint does not work in this case. A different approach has to be used.

4.2 Closed Curves

The closedness issue of a fine tuned subdivision curve is discussed in this subsection. To ensure that a closed subdivision curve remains closed after the fine tuning process, one needs the capability of restructuring the control polygon of the curve geometrically (but not topologically) without inducing large change on the curvature profile of the resulting, fine-tuned curve. This can be achieved by assuming that legs of the control polygon are made of flexible springs and adjacent legs are joined with rotational springs. The control points of the fine-tuned curve are

determined by minimizing the following energy function E:

$$E = \sum_{i=1}^{m} \Big(k_l (l_i^{opt} - l_i)^2 + k_a (\theta_i^{opt} - \theta_i)^2 \Big), \tag{13}$$

where m is the number of polygon legs which is equal to the number of joints, and k_l and k_a are constants called *spring coefficients*. l_i is the length of the i-th leg and θ_i is the joint angle between the $(i-1)$-st and i-th legs. l_i^{opt} and θ_i^{opt} are the optimal lengths and joint angles of the corresponding legs. In the optimal case the start and the end points of the fine-tuned curve coincide. Therefore we assume l_i^{opt} and θ_i^{opt} are the lengths and joint angles of the fine-tuned curve without the end-point constraint. Because of the invariance property of the endpoint tangent directions, the joint angle between the first and the last legs can be obtained simply by calculating the angle between them. The position of at least one point and rotation about at least one point of the curve should be fixed to avoid rigid motion during the optimization.

The conjugate gradient method (see, for instance, [13]) is used for the minimization process. The initial values of the minimization process are the control points of the invariant curve fine tuned by an identity scalar function whose degree is 1 and whose segmentation equals that of the given scalar function. To achieve geometrically similar effects under uniform scaling, the original curve is pre-scaled to fit in a unit square. After the fine tuning process, the curve will be rescaled to the original size. The final shape also depends on the ratio of the spring coefficients k_l and k_a. Large k_a relative to k_l implies a stronger resistance to change the joint angles and, therefore, a bigger tendency for the fine-tuned curve to preserve the local maxima of the curvature. The ratio used in the examples in Figures 14(c) and 14(e) is $k_l : k_a = 1 : 2$. The white marks on the curves in these figures indicate the fixed points. The processing time for the deformation is less than 0.1 second. The deformation performed by elongating the bottom leg of the original curve is shown in Figure 14(f) as a comparison to our method. Because of the limited degree of freedom of the curve, it has a small curvature around the top.

4.3 Subdivision surface

The three issues (domain, product and closedness) that have to be addressed when dealing with closed subdivision curves and surfaces are more complicated in the surface case. Fortunately, they can still be resolved using the same strategy as the curve case.

The key idea in applying the fine tuning technique to subdivision surfaces is that segmentation of the parameter space of the scalar function α must have the same knot intervals as the original subdivision surface, so that one only need to assign control values to the edges or vertices of the control mesh.[4] This requirement puts some restriction on the choice of the scalar function, but would still leave one with enough room to yield various deformations, as will be seen in this section. It should be pointed out that, by assigning control values to the edges of a subdivided control mesh instead of the original control mesh, one can overcome this restriction to have a finer segmentation for the scalar function.

The Doo-Sabin and Catmull-Clark surfaces are two popular subdivision surfaces used in graphics community. They are constructed by generalizing the idea of obtaining uniform biquadratic and uniform bicubic B-spline patches from a rectangular mesh, respectively. When using a Doo-Sabin surface as the original surface and applying a linear scalar function to it for the fine tuning process, the ideal situation would be for the fine-tuned surface to become a Catmull-Clark surface. This is a desirable combination of the degrees of the original surface and the fine-tuned surfaces as far as continuity is concerned, as mentioned in Section 2. As the degree elevation process requires the use of multiple knots, the uniform Catmull-Clark formulation can not represent a Doo-Sabin surface even if it is defined by a regular control mesh. Fortunately, non-uniform versions of these surfaces have been presented by Sederberg et al. [16] and a Doo-Sabin surface can be converted to a non-uniform Catmull-Clark surface precisely for a regular control mesh and approximately for a control mesh with arbitrary topology, as explained in Appendix (b).

Figure 16(b) shows a typical Doo-Sabin surface defined by the control mesh shown in 16(a). The control mesh is converted to the control mesh of a non-uniform Catmull-Clark surface in 16(c) where 1 is assigned as the knot interval to each edge in yellow color and 0 to each edge in red.

[4] Whether the control values are assigned to the edges or vertices depends on the degree of the scalar function. If the scalar function is linear or cubic, the control values are assigned to the edges. If the scalar function is quadratic, the control values are assigned to the vertices. This is because each segment of the scalar function corresponds to an edge or vertex of the control mesh.

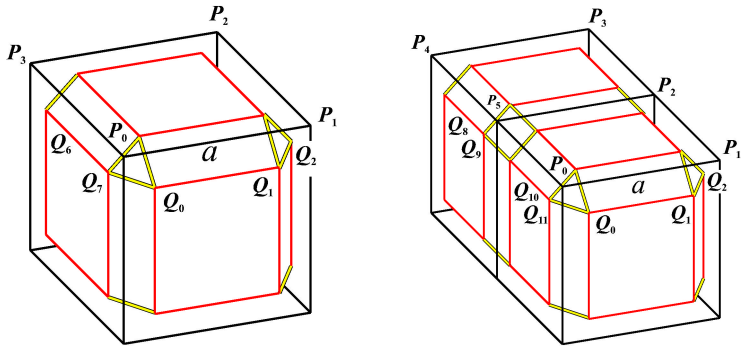
4.4 Closed surface

A physical model similar to the curve case is used to keep the fine-tuned surface closed. As a preliminary process, each edge of the control mesh is assigned a control value a_i as a specified value of the scalar function α. Each edge is again assumed to be made of a flexible spring and two adjacent edges connected to the same vertex are jointed with a rotational spring. An energy function E similar to Eq.(13) is defined and minimized to determine the positions of the control points of the fine-tuned surface:

$$E = k_l \sum_{i=1}^{m_0} (l_i^{opt} - l_i)^2 + k_a \sum_{i=1}^{m_1} (\theta_i^{a,opt} - \theta_i^a)^2 + k_b \sum_{i=1}^{m_2} (\theta_i^{b,opt} - \theta_i^b)^2, \tag{14}$$

where m_0 is the number of edges of the control mesh, m_1 is the number of constrained pairs of edges connected to the same vertices, as will be explained later in this subsection, and m_2 is the number of total pairs of adjacent edges connected to the same vertices, i.e. the total number of the valences (degrees) of all the vertices of the control mesh. l_i^{opt} and $\theta_i^{a,opt}$ are defined as in Section 4.2. The first and second summations are responsible for the fine tuning process and the last term is for preserving original shape. The calculation of the first term is straightforward if the optimal lengths l_i^{opt} of the edges are available. The second term is calculated as follows. Consider, for example, the boundary edges of the face $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$ of the Doo-Sabin control mesh in Figure 15(a). These edges are transformed to an edge loop $\boldsymbol{Q}_0\boldsymbol{Q}_1 \cdots \boldsymbol{Q}_7$ in the converted Catmull-Clark control mesh. By using the assigned control values of the edges (value a is assigned to edge $\boldsymbol{P}_0\boldsymbol{P}_1$ in both (a) and (b)) we can determined optimal lengths l_i^{opt} of the edges and optimal angles $\theta_i^{a,opt}$ of the edge joints in the loop. Hence we can obtain the optimal length of, for instance, edge $\boldsymbol{Q}_0\boldsymbol{Q}_1$ and the optimal joint angle between edges $\boldsymbol{Q}_7\boldsymbol{Q}_0$ and $\boldsymbol{Q}_0\boldsymbol{Q}_1$. The situation could be more complicated and it is possible that the boundary edges of several faces correspond to the same edge loop. For example, in Figure 15(b), the boundaries of the two faces $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_5$ and $\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{P}_4\boldsymbol{P}_5$ correspond to the same edge loop $\boldsymbol{Q}_0\boldsymbol{Q}_1 \cdots \boldsymbol{Q}_{11}$. Identifying the loops which are the targets of the fine tuning is not difficult because the degree of each vertex of the converted control mesh is equal to 4 and a loop is identified by tracing the opposite edge each time we enter a new vertex. It is easy to see that each edge of the converted polygon belongs to one and only one loop. For the shape preservation term, $\theta_i^{b,opt}$ is calculated from the converted control mesh.

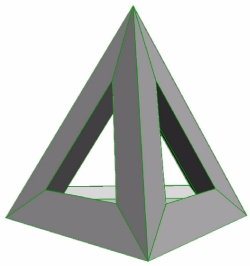
Figure 16(f) shows an example of a fine-tuned Doo-Sabin subdivision surface. Initially, each edge of the control mesh is assigned a value 1 as the default control value. Some of the edges are then given different values to deform the surface. To prevent rigid motion and to reduce the number of parameters in the optimization process, one may have some of the vertices fixed. For this example, 1.3 is assigned to each of the three edges connected to the top vertex as the control value and three vertices at the bottom are fixed. The conjugate gradient method is used for the minimization process as in the curve case. The original surface is pre-scaled to fit in a unit cube and the ratio $k_l : k_a : l_b = 1 : 1 : 1$ is used. The processing time for the deformation is less than a second. The yellow edges in 16(b) and 16(c) are given the value of 1 as their knot intervals. The red edges are given 0.05 instead of the theoretical value of 0 to clarify the effect of small knot interval value which generates dense meshes around. The example shows that one can deform a subdivision surface without even manipulating its control mesh.



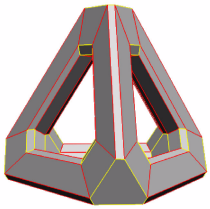
(a) Edge loop around one face.

(b) Edge Loop around two faces.

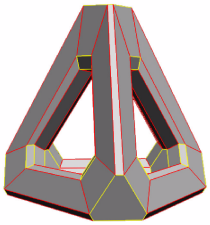
Figure 15. Extraction of edge loop.



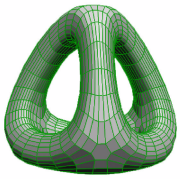
(a) Original control mesh.



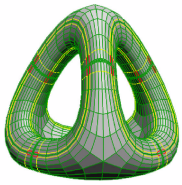
(b) Converted control mesh.



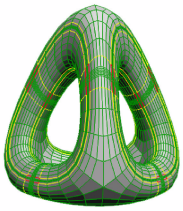
(c) Deformed control mesh.



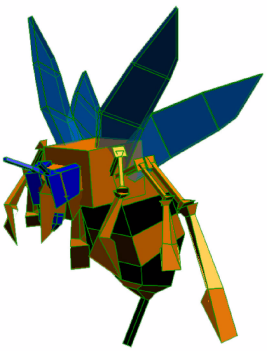
(d) Doo-Sabin surface (depth=3).



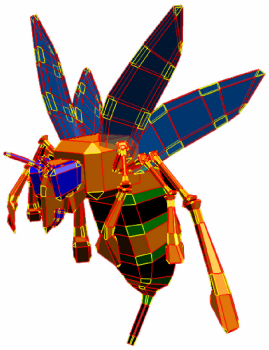
(e) Non-uniform Catmull-Clark surface (depth=3).



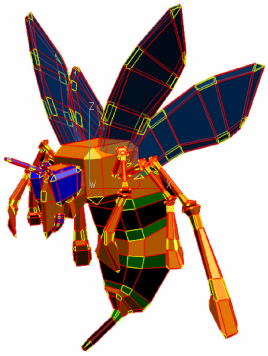
(f) Deformed non-uniform Catmull-Clark surface (depth=3).



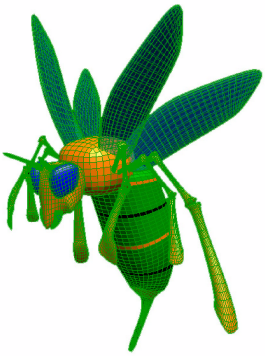
(g) Original control mesh.



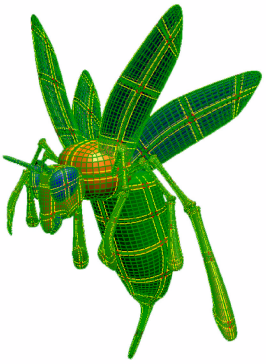
(h) Converted mesh.



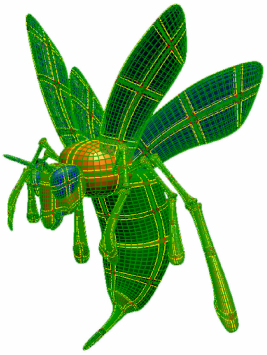
(i) Deformed control mesh of non-uniform Catmull-Clark surface. The belly and all the wings are deformed.



(j) Doo-Sabin surface (depth=3).



(k) Non-uniform Catmull-Clark surface (depth=3).



(l) Deformed non-uniform Catmull-Clark surface (depth=3).

Figure 16. Examples of subdivision surface fine tuning.

Figures from 16(g) to 16(l) show another example of fine tuning a Doo-Sabin subdivision surface. Figure 16(l) shows a fine-tuned surface defined by the control mesh in 16(i). The belly and all the wings of the wasp model are deformed. The processing times of the deformations are from a second to 10 seconds, depending on the number of parameters. The example further shows that our fine tuning technique has the power to deform a portion of a complicated object.

To use the fine tuning technique to deform a Catmull-Clark surface, a quadratic scalar function is recommended and the fine-tuned surface should be a subdivision surface based on the biquintic tensor product surface.

5 Conclusion

A new deformation-based fine tuning technique for parametric curves and surfaces has been proposed and discussed. The new approach is different from traditional approaches in that the deformation is performed by scaling the derivative of the curve or surface, instead of manipulating its control points. Therefore, the new approach allows direct manipulation of the curvature (and, consequently, fairness) of a curve or surface. The new technique allows fine tuning of a curve or surface without changing the basic shape of its profile and curvature distribution. Precise shaping and deformation, such as making the curvature of a region twice as big, is possible with the new approach. We have shown how to use the new technique to perform fine tuning on an arbitrary portion of a curve or surface. We have also shown how to use the new technique to fine tune subdivision curves and surfaces.

This is the first time a deformation technique based on scaling the derivative of a curve or surface is proposed. To ensure that this is indeed a powerful fine tuning technique, we have tested the new technique on various data sets with various different requirements. In addition to the applications cited above, we also see the possibly of using the new technique for multi-resolution deformation of curves and surface and wavelet analysis. Another possible application area is the fine tuning of implicit curves and surfaces. A difficult task in that area is the selection of differentiation direction for surfaces.

References

[1] E. Catmull, and J. Clark, "Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes," *Computer-aided Design*, Vol.10, No.6, pp.350-355, 1978.

[2] G. Chaikin, "An Algorithm for High-Speed Curve Generation," *Computer Graphics and Image Processing*, No.3, pp.346-349, 1974.

[3] T. DeRose, M. Kass, and T. Truong, "Subdivision Surfaces in Character Animation," *Computer Graphics* (*Proc. of SIGGRAPH '98*), Vol. 32, pp.85-94, July, 1998.

[4] D. Doo, and M. Sabin, "Behaviour of Recursive Division Faces Near Extraordinary Points," *Computer-aided Design*, Vol.10, No.6, pp.356-360, 1978.

[5] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*, 4th Ed., Academic Press, 1997.

[6] D.S. Meek, and R.S.D. Thomas, "A Guided Clothoid Spline," *Computer Aided Geometric Design*, Vol.8, pp.163-174, 1991.

[7] K.T. Miura, "Unit Quaternion Integral Curve: A New Type of Fair Free-form Curves," *Computer Aided Geometric Design*, Vol.17, No.1, pp.39-58, 2000.

[8] H.P. Moreton, and C.H. Séquin, "Functional Optimization for Fair Surface Design," *Computer Graphics* (*Proc. of SIGGRAPH '92*), Vol.26, No.2, pp.167-176, 1992.

[9] K. Morken, "Some Identities for Products and Degree Raising of Splines," *Constructive Approximation*, Vol. 27, No.2, pp.195-208, 1991.

[10] L. Piegl, and W. Tiller, "Algorithm for Degree Reduction of B-spline Curves." *Computer-aided Design*, Vol.27, No.2, pp.101-110, 1995.

[11] L. Piegl, and W. Tiller, *The NURBS Book*, 2nd Ed. Springer-Verlag, 1997.

[12] L. Piegl, and W. Tiller, "Symbolic Operators for NURBS." *Computer-aided Design*, Vol.29, No.5, pp.361-368, 1997.

[13] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988.

[14] N.S. Sapidis, ed., *Designing Fair Curves and Surfaces*, SIAM, 1994.

[15] T. W. Sederberg and S. R. Parry, "Free-Form Deformation of Solid Primitives," *Computer Graphics* (*Proc. of SIGGRAPH'86*), Vol. 20, pp.151-160, August, 1986.

[16] T.W. Sederberg, J. Zheng, D. Sewell and M. Sabin, "Non-Uniform Recursive Subdivision Surfaces," *Computer Graphics* (*Proc. of SIGGRAPH'98*) Vol. 32, pp.387-394, July, 1998.

[17] D.J. Walton, and D.S. Meek, "Clothoidal Splines," *Computers and Graphics*, Vol.14, No.1, pp.95-100, 1990.

Appendix

(a) Degree elevation

The degree elevation of a B-spline curve is shown here. We follow the notations of Piegl and Tiller [11] in this work.

Let $\boldsymbol{C}_p = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P}_i$ be an end point interpolating (nonperiodic) degree p B-spline curve with respect to the knot vector U. To elevate its degree to $p+1$, one needs to construct a knot vector $\hat{U}$ and control points $\boldsymbol{Q}_i$ so that

$$\boldsymbol{C}_p(u) = \boldsymbol{C}_{p+1}(u) = \sum_{i=0}^{\hat{n}} N_{i,p+1}(u)\boldsymbol{Q}_i. \tag{15}$$

Assume U has the following form:

$$U = \{u_0, \cdots, u_m\} = \{a, \cdots, a, u_1, \cdots, u_1, \cdots, u_s, \cdots, u_s, b, \cdots, b\} \tag{16}$$

where the multiplicities of the interior knots are $m_1, \cdots, m_s$, respectively. At a knot of multiplicity m_i, $\boldsymbol{C}_p(u)$ is C^{p-m_i} continuous and $C_{p+1}(u)$ must have the same degree of continuity there. Therefore, $\hat{n} = n + s + 1$ and

$$\hat{U} = \{u_0, \cdots, u_{\hat{m}}\} = \{a, \cdots, a, u_1, \cdots, u_1, \cdots, u_s, \cdots, u_s, b, \cdots, b\} \tag{17}$$

where $\hat{m} = m + s + 2$. The control points $\boldsymbol{Q}_i$ of the degree-elevated curve $C_{p+1}(u)$ are determined by solving the following system of linear equations:

$$\sum_{i=0}^{\hat{n}} N_{i,p+1}(u_j)\boldsymbol{Q}_i = \sum_{i=0}^{n} N_{i,p}(u_j)\boldsymbol{P}_i, \qquad j = 0, \cdots, \hat{n}, \tag{18}$$

where u_j are $\hat{n}+1$ appropriate parameter values. The degree elevation of a NURBS curve can be done similarly and there are more efficient methods to calculate $\boldsymbol{Q}_i$ [11].

In case of a periodic curve, if the knot vector U is $\{u_0, u_1, \cdots, u_m\}$, then $\hat{U}$ is given by

$$\hat{U} = \{u_1, u_1, \cdots, u_{m-1}, u_{m-1}\}. \tag{19}$$

For example, the closed quadratic B-spline curve constructed from the triangular control polygon in Figure 14(a) consists of three segments and its knot vector U is as follows:

$$U = \{-\frac{2}{3}, -\frac{1}{3}, 0, \frac{1}{3}, \frac{2}{3}, 1, \frac{4}{3}, \frac{5}{3}\}. \tag{20}$$

The knot vector $\hat{U}$ and the control points $\boldsymbol{Q}_i$ of the degree elevated cubic B-spline curve are given by

$$U = \{-\frac{1}{3}, -\frac{1}{3}, 0, 0, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, 1, 1, \frac{4}{3}, \frac{4}{3}\}, \tag{21}$$

and

$$\boldsymbol{Q}_{2i} = \frac{5}{6}\boldsymbol{P}_i + \frac{1}{6}\boldsymbol{P}_{i+1}, \boldsymbol{Q}_{2i+1} = \frac{1}{6}\boldsymbol{P}_i + \frac{5}{6}\boldsymbol{P}_{i+1}, \tag{22}$$

for $i = 0, \cdots, 3$. The original quadratic B-spline curve and the degree elevated curve can be regarded as a uniform and a non-uniform B-spline subdivision curve, respectively. The knot intervals of the degree elevated curve are shown in Figure 14(d).

(b) Approximate conversion from a Doo-Sabin subdivision surface to a non-uniform Catmull-Clark subdivision surface

The approximate conversion from a Doo-Sabin subdivision surface to a non-uniform Catmull-Clark subdivision surface can be performed topologically in the same way as the Doo-Sabin subdivision, but geometrically they are different and different subdivision coefficients are used for the conversion process. We apply to the Doo-Sabin surface a linear B-spline function whose parameter domain is identical to that of the original surface.

For a regular mesh, as the one shown in Figure 17(a), the conversion may be regarded as a degree elevation of a uniform biquadratic surface to a non-uniform bicubic B-spline surface. In the figure, the green lines are patch boundaries of the

quadratic B-spline surface. The yellow and red lines are those of the non-uniform cubic B-spline surface whose knot intervals are 1 and 0, respectively. The new control points $\boldsymbol{F}_a$ is given by

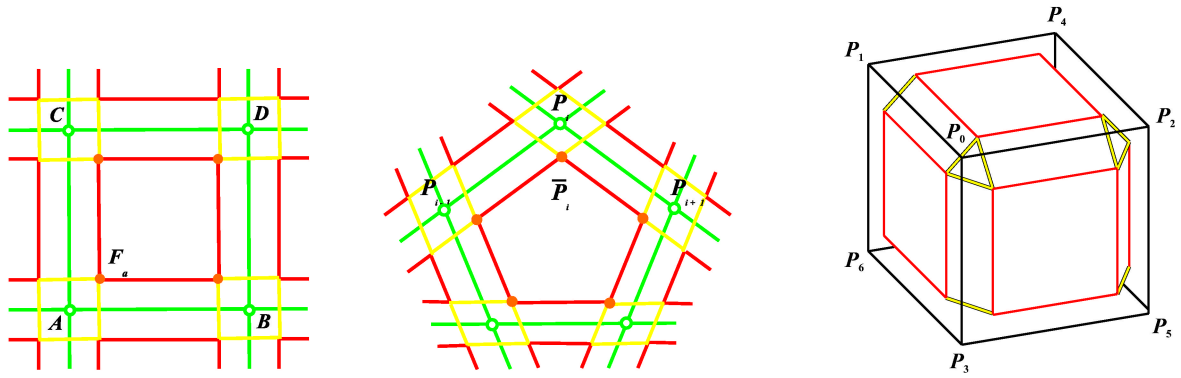
$$\boldsymbol{F}_a = \frac{25}{36}\boldsymbol{A} + \frac{5}{36}(\boldsymbol{B}+\boldsymbol{C}) + \frac{1}{36}\boldsymbol{D} = \frac{(\boldsymbol{V}+2\boldsymbol{A})}{3} + \frac{\boldsymbol{B}+\boldsymbol{C}-\boldsymbol{A}-\boldsymbol{D}}{9}, \tag{23}$$

where $\boldsymbol{V} = (\boldsymbol{A}+\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{D})/4$.

The n-sided faces where $n \neq 4$ remain n-sided after the conversion in a manner identical to the Doo-Sabin subdivision. In Figure 17(b), the new vertex $\bar{\boldsymbol{P}}_i$ is calculated by

$$\bar{\boldsymbol{P}}_i = (\frac{4}{9} + \frac{1}{n})\boldsymbol{P}_i + \frac{1}{9n}\sum_{j=1,j\neq i}^{n}(5 + 4\cos(\frac{2\pi|i-j|}{n}))\boldsymbol{P}_j. \tag{24}$$

The yellow and red lines are boundaries of the non-uniform cubic B-spline surface whose knot intervals are 1 and 0, respectively, as the regular mesh case.



(a) Regular mesh. (b) Non-four-sided face. (c) Conversion error.

Figure 17. The conversion process and its error.

For a regular mesh, the above conversion is exact in the sense that a Doo-Sabin subdivision surface is exactly converted to a non-uniform Catmull-Clark subdivision surface. However, for an irregular mesh, the conversion is approximate. For example, the point $\boldsymbol{P}_0$ of the Doo-Sabin control mesh in Figure 17(c) will converge to $\boldsymbol{P}_0^\infty$:

$$\boldsymbol{P}_0^\infty = \frac{9}{16}\boldsymbol{P}_0 + \frac{1}{8}(\boldsymbol{P}_1+\boldsymbol{P}_2+\boldsymbol{P}_3) + \frac{1}{48}(\boldsymbol{P}_4+\boldsymbol{P}_5+\boldsymbol{P}_6). \tag{25}$$

The corresponding point $\boldsymbol{Q}_0^\infty$ of the converted Catmull-Clark surface is given by

$$\boldsymbol{Q}_0^\infty = \frac{629}{1152}\boldsymbol{P}_0 + \frac{223}{172}(\boldsymbol{P}_1+\boldsymbol{P}_2+\boldsymbol{P}_3) + \frac{77}{3456}(\boldsymbol{P}_4+\boldsymbol{P}_5+\boldsymbol{P}_6). \tag{26}$$

The difference of these two points is

$$\boldsymbol{P}_0^\infty - \boldsymbol{Q}_0^\infty \approx 0.0165\boldsymbol{P}_0 - 0.00405(\boldsymbol{P}_1+\boldsymbol{P}_2+\boldsymbol{P}_3) - 0.00145(\boldsymbol{P}_4+\boldsymbol{P}_5+\boldsymbol{P}_6). \tag{27}$$

This is small enough for practical CG applications where extremely high degree is not the primary concern. The primary concern is the deformation process.