NSF GRANT #0422126
NSF PROGRAM NAME: Integration Engineering

Locally Adjustable Interpolation for Meshes of Arbitrary Topology

Lead P.I. Fuhua (Frank) Cheng
University of Kentucky

Abstract: A new method for constructing a smooth surface that interpolates the vertices of an arbitrary mesh is presented. The mesh can be open or closed. Normals specified at vertices of the mesh can also be interpolated. The interpolating surface is obtained by locally adjusting the limit surface of the given mesh (viewed as the control mesh of a Catmull-Clark subdivision surface) so that the modified surface would interpolate all the vertices of the given mesh. The local adjustment process is achieved through locally blending the limit surface with a surface defined by non-uniform transformations of the limit surface. This local blending process can also be used to smooth out the shape of the interpolating surface. Hence, a *surface fairing* process is not needed in the new method. Because the interpolation process does not require solving a system of linear equations, the method can handle meshes with large number of vertices. Test results show that the new method leads to good interpolation results even for complicated data sets. The new method is demonstrated with the Catmull-Clark subdivision scheme. But with some minor modification, one should be albe to apply this method to other subdivision schemes as well.

1. Introduction: Constructing a smooth surface to interpolate the vertices of a given mesh is an important task in many areas, including geometric modeling, computer graphics, computer animation, interactive design, and scientific visualization. The interpolating surface sometime is also required to interpolate normal vectors specified for some or all of the mesh vertices. Developing a general solution for this task is difficult because the required interpolating surface could be of arbitrary topology and with arbitrary genus. Traditional representation schemes such as B-spline or Bézier surfaces can not represent such a complex shape with only one surface.

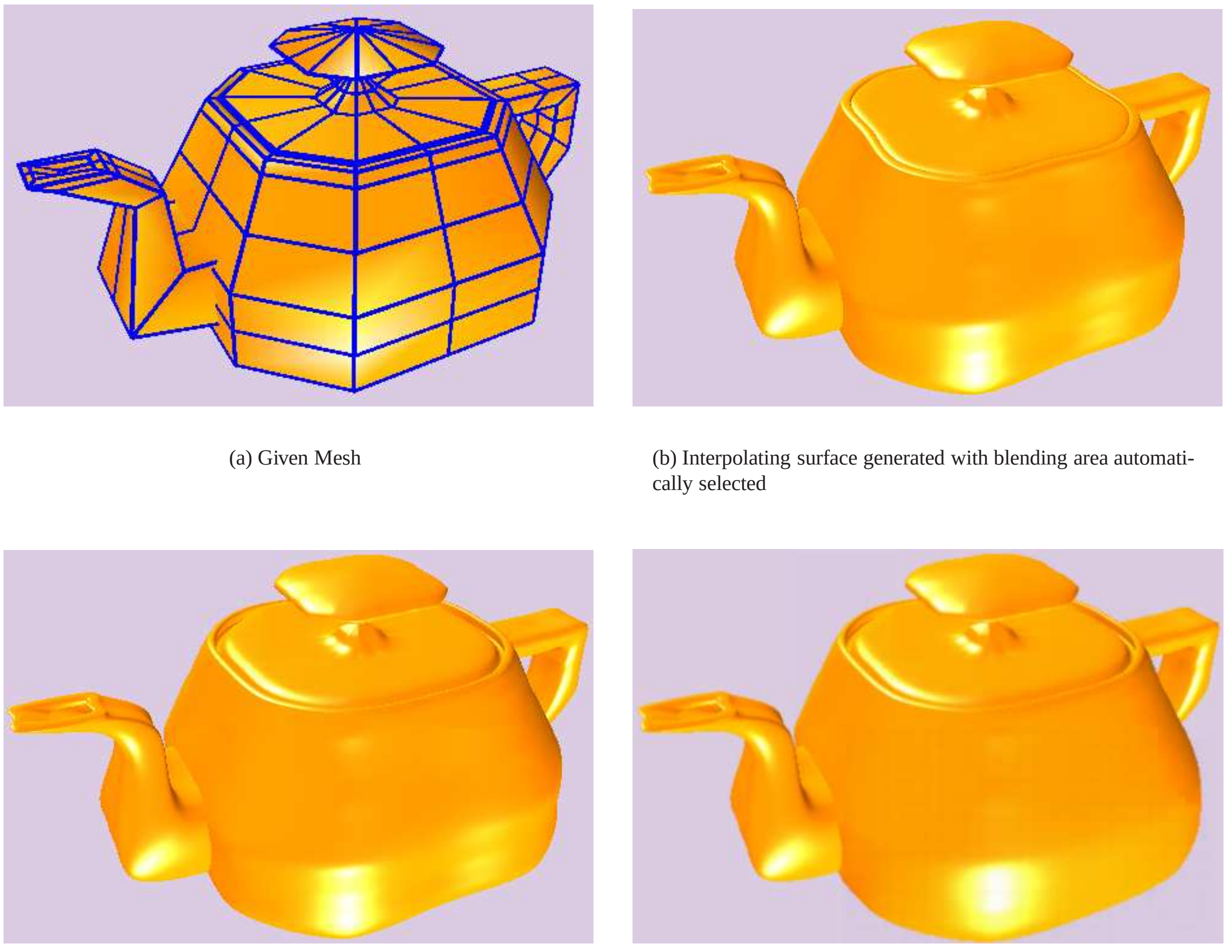
Subdivision surfaces were introduced as an efficient technique to model complex shapes [2][3][10]. But building a connection between a given mesh and an interpolating subdivision surface has never really been successful when the number of vertices of the given mesh is large [1]. One exception is a work published recently [11]. In this paper, an iterative interpolation technique similar to the one used in [8] for non-uniform B-spline surfaces is proposed for subdivision surfaces. Since the iterative approach does not require solving a system of linear equations, it can handle meshes with large number of vertices. But the paper fails to prove the convergence of the iterative process.

In this paper we will address the problem of 'constructing a smooth surface to interpolate the vertices of a given mesh' and present a new solution to this problem. We briefly review previous work in this area first.

1.1. Previous Work: A Brief Review: There are two major ways to interpolate a given mesh with a subdivision surface: *interpolating subdivision* [4, 6, 7, 14, 19] or *global optimization* [5, 12]. In the first case, a subdivision scheme that interpolates the control vertices, such as the Butterfly scheme [4], Zorin et al's improved version [19] or Kobbelt's scheme [7], is used to generate the interpolating surface. New vertices are defined as local affine combinations of nearby vertices. This approach is simple and easy to implement. It can handle meshes with large number of vertices. However, since no vertex is ever moved once it is computed, any distortion in the early stage of the subdivision will persist. This makes interpolating subdivision very sensitive to irregularity in the given mesh. In addition, it is difficult for this approach to interpolate normals or derivatives.

The second approach, *global optimization*, usually needs to build a global linear system with some constraints [13]. The solution to the global linear system is a control mesh whose limit surface interpolates the vertices of the given mesh. This approach usually requires some

[1]Interpolating subdivision [4] will be addressed shortly


(a) Given Mesh

(b) Interpolating surface generated with blending area automatically selected

(c) Interpolating surface generated with user selected blending areas around upper portion of the teapot body

(d) Interpolating surface generated with user selected blending areas around bottom portion of the teapot body

Figure 1: Example with local control

fairness constraints in the interpolation process, such as the energy functions presented in [5], to avoid undesired undulations. Although this approach seems more complicated, it results in a traditional subdivision surface. For example, the method in [5] results in a Catmull-Clark subdivision surface (CCSS), which is C^2-continuous almost everywhere and whose properties are well studied and understood. The problem with this approach is that a global linear system needs to be built and solved. It is difficult for this approach to handle meshes with large number of vertices.

There are also techniques that produce surfaces to interpolate given curves or surfaces that near- (or quasi-) interpolate given meshes [9]. But those techniques are either of different natures or of different concerns and, hence, will not be discussed here.

1.2. Overview: In this paper a new method for constructing a smooth surface that interpolates the vertices of a given mesh is presented. The mesh can be of arbitrary topology and can be open or closed. Normal vectors specified for any vertices of the mesh can also be interpolated. The basic idea is to view the given mesh as the control mesh of a Catmull-Clark subdivision surface and locally adjust the limit surface of the given mesh so that the resulting surface would not only interpolate vertices of the given mesh, but also possess a satisfactory smooth shape. The local adjustment process is achieved through blending the limit surface S with a blending surface T defined by non-uniform transformations of the limit surface. By performing the blending process at different selected points, we are able to (1) ensure the modified surface would interpolate the given mesh, (2) prevent it from generating unnecessary undulations, and (3) smooth out the shape of the resulting surface.

The new method has two main advantages. First, since we do not have to compute the interpolating surface's control mesh, there is no need to solve a system of linear equations. Therefore, the new method can handle meshes with large number of vertices, and is more robust and stable. Second, because the local blending process can be used to smooth out the shape of the interpolating surface, a *surface fairing* process is not needed in the new method.

An example of this interpolation process is shown in Figure 1. The surfaces shown in Figures 1(b), 1(c) and 1(d) all interpolate the mesh shown in Figure 1(a). The blending areas in Figure 1(b) are automatically selected by the system while Figures 1(c) and 1(c) have user selected blending areas in the upper portion and lower portion of the teapot body afterward. It is easy to see from Figure 1 that local control is necessary when better quality interpolating surfaces are needed.

The new method is demonstrated with Catmull-Clark subdivision surfaces here (by viewing the given mesh as the control mesh of a Catmull-Clark subdivision surface). But with a minor modification, one should be able to apply it to other subdivision schemes as well.

The remaining part of the paper is arranged as follows. In Section 2, the basic idea of our locally controllable interpolation technique for closed meshes is presented. The construction process of a blending surface is presented in Section 3. In Section 4, a local parametrization is introduced. The blending process around an extraordinary point or an arbitrarily selected point is discussed in Section 5 and Section 6, respectively. Issues on dealing with normal interpolation and handling open meshes are discussed in Section 7 and Section 8, respectively. Implementation issues and test results are presented in Section 9. Concluding marks are given in Section 10.

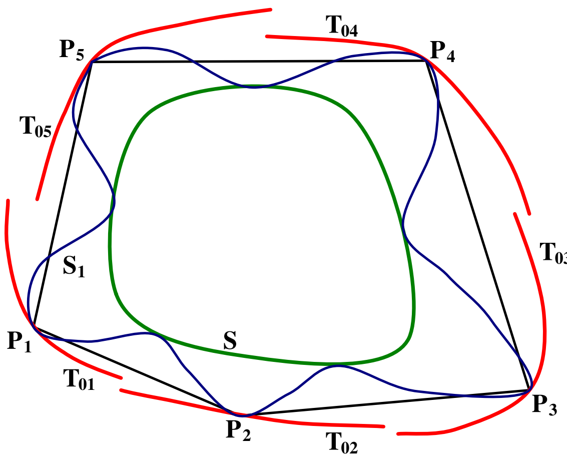
2. Basic Idea: Given a 3D mesh with n vertices: $P = \{\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_n\}$, the goal here is to construct a new surface that interpolates P (the vertices of P, for now). Contrast to existing interpolation methods, which either construct a new mesh whose limit surface interpolates P or perform interpolating subdivision schemes on the input mesh, we perform interpolation by manipulating the limit surface S of the given mesh directly. The basic idea is to *push* or *pull* the limit surface of the given mesh in vicinity of selected points so that the modified surface interpolates the given mesh and, in the meanwhile, prevent it from generating unnecessary undulations and maintain its smoothness. The push or pull process is done by constructing a new surface T, and blending T with S. T must be relatively easy to construct and interpolating P initially. For example, in Fig. 2(a), T is composed of five separate segments: T_{01}, T_{02}, T_{03}, T_{04} and T_{05}, and each of them interpolates a point of $P = \{P_1, P_2, P_3, P_4, P_5\}$. T and S must be blended in a way such that the resulting surface interpolates P and is C^2-continuous almost everywhere. The interpolating surface can be defined as follows:

$$\bar{S} = S(u,v)W(u,v) + T(u,v)(1 - W(u,v)), \quad (1)$$

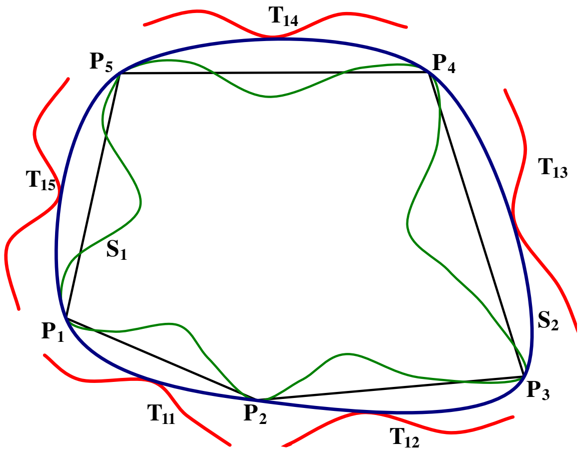
where $0 \leq W(u,v) \leq 1$ is a C^2-continuous weight function satisfying the property $\lim_{(u,v)\to 0} W(u,v) = 0$. The blending process is done independently on each of the three coordinates of the surface $S(u,v)$. T must be parametrized so that $T(0,0) = P_i$, $(1 \leq i \leq n)$ and is C^2-continuous everywhere except at $(0,0)$ (where it is at least C^1-continuous) and except at $\{(u,v) \mid W(u,v) = 1\}$ (where it is not even necessary to be C^0-continuous). Therefore $\bar{S}$ is guaranteed to interpolate P and is C^2-continuous everywhere except at some extraordinary points.

Usually during the initial blending process, quality of the resulting interpolating surface would not be good enough yet. For example, the blue curve in Fig. 2(a), denoted S_1, is the resulting curve of the first blending process. As we can see, S_1 has a lot of undesired undulations although it interpolates the given mesh P exactly.

To improve the shape of the interpolating surface and to reduce unnecessary oscillations, a second blending process can be performed in the vicinity of some selected points. For example, in Fig. 2(b), a second blending process is performed in the vicinity of all the *edge points* of the given mesh. To carry out the second blending process, a different blending surface T_1 has to be constructed. T_1 does not have to interpolate P. However, T_1 must not change the position of the images of P on the limit surface. In other words, the domain involved in constructing T_1 should be smaller than the domain of S_1, so that the images of P would not be involved in the construction process of T_1. For example, in Fig. 2(b),



(a) First blending



(b) Second blending

Figure 2: Basic idea of the new interpolation method.

$T_1 = \{T_{11}, T_{12}, T_{13}, T_{14}, T_{15}\}$ and the images of P_is are not involved in the construction of T_1. Once T_1 is constructed, T_1 can be blended with S_1 similarly to get S_2 as follows:

$$S_2 = S_1(u,v)W_1(u,v) + T_1(u,v)(1 - W_1(u,v)),$$

where $W_1(u,v)$ is a blending function similar to $W(u,v)$ in Eq. (1), except $W_1(u,v)$ is constructed for vicinity of edge points, while $W(u,v)$ is constructed for vicinity of vertex points. This means that we have to translate (u,v) by some constant so that $W_1(u,v) = 0$ at the selected edge point. Because the images of P are not involved in the construction process of T_1, the images of P are not affected in the above blending process. Hence interpolation requirement still holds.

Note that the blending process is done for individual pieces. For example, in Fig. 2(b), it is done for the pieces corresponding to $T_{1i},\ 1 \le i \le 5$, independently. Because T_1 is not required to interpolate P, not every $T_{1i},\ 1 \le i \le 5$, has to be blended with the corresponding piece of S_1. A blending process is performed for a selected region only if the shape of the surface is not good enough in that area. Hence, the blending process is an optional operation.

As we can tell from Fig. 2, the shape of S_2 is much better than that of S_1. However, if necessary, a third or even more blending processes can be performed on the resulting surface to further improve its quality. While the above idea seems to be simple and straightforward, the key here is how to construct a $T(u,v)$ for each local blending process and how to construct the corresponding blending weight function $W(u,v)$ such that the resulting interpolating surface is smooth and oscillation-free. The construction process of T will be shown in the next section. For consistency, we denote the $(i+1)$st blending surface T by T_i, and use T as a general reference to all possible levels of T_i.

3. Construction of T: The construction process of T must satisfy two requirements: it should be intuitive enouch to use and the result should be easy to obtain. Note that only T_0 is required to interpolate P, not the subsequent $T_i,\ i \ge 1$. Hence, it is sufficient to show the construction process of T_0 only. T_i $(i \ge 1)$ can be constructed similarly, without the interpolation constraint. Nevertheless, we will show how to construct T_i $(i \ge 1)$ with details in Section 6 after local parameterization of subdivision surfaces is discussed.

T can be constructed in several different ways. In this paper we construct T by linearly transforming pieces of S in 3D object space. Note that T_0 is not necessary to be C^0-continuous at parameters where $W(u,v) = 1$. The affine transformation matrix can be chosen in a way such that T_0 interpolates P and, in the meanwhile, changes the original limit surface as little as possible. For example, in Figure 2(a), to get T for the limit surface S defined by $P = \{P_1, P_2, P_3, P_4, P_5\}$, we simply translate S segment by segment, in the direction from the image of P_i to

P_i, and then scale each segment with appropriate scaling factors for X, Y and Z components such that T interpolates P and has an appropriate size. Consequently, as it shows in Figure 2(a), T is represented by five segments: $\{T_{01}, T_{02}, T_{03}, T_{04}, T_{05}\}$ and they are not C^0-continuous. But after a blending process using equation (1), we get a surface S_1 which smoothly interpolates P.

4. Local Parameterization of Subdivision Surfaces: The blending process defined by eq. (1) is performed on regions of the limit surface. Hence a local parameterization is needed for each region of the limit surface where a blending process is to be performed. Several local parameterization methods have been reported in the literature [15, 17]. We follow Reif's approach [17] here. We assume that for any two extraordinary points of P, their corresponding points on the limit surface are at least two patches away. If this is not the case, simply perform one or two subdivision steps on P to get a new control mesh for the limit surface. But the target of the interpolation process is still P, not the new control mesh.

Reif's approach maps an extraordinary point to $(0, 0)$ and is based on the *characteristic map* of a subdivision scheme [17]. A characteristic map is defined by calculating the limit of subdivision on a 2D mesh formed by the two sub-dominant eigenvectors of the local subdivision matrix [17, 21]. The characteristic map for Catmull-Clark subdivision scheme around an extraordinary vertex of valence n is based on the topology of the 2-ring neighborhood of vertices around the extraordinary vertex. The 2-ring neighborhood is enough to determine the limit function for the n faces adjacent to the extraordinary vertex. Thus, the two sub-dominant eigenvectors have $6n + 1$ entries each. Since they do not depend on the input mesh, they can be pre-computed for each valence n. Once we have the two sub-dominant eigenvectors, we can find (u, v) parameters corresponding to each vertex of the k-times ($k \in Z$ and $k \geq 0$) refined mesh, around the extraordinary point. This is done by applying a 3×3 limit mask [18] of bicubic B-splines to the corresponding neighborhood of vertices in the k-times refined mesh. Also as a normalization rule, the two sub-dominant eigenvectors should be scaled such that the parameters (u, v) at the end-points of edges emanating from the extraordinary vertex have coordinates $(\cos(i\alpha), \sin(i\alpha))$, $i = 1 \cdots n$, where $\alpha = 2\pi/n$ (see Fig. 3(a)).

5. Blending around an extraordinary point: With parametrization available, it is now possible to perform blending process on regions of the limit surface. To maximize the blending area around an extraordinary point (note that a regular point is just a special case of an extraordinary point), we define the blending region in the parameter space by the condition:

$$u^2 + v^2 \leq 1.$$

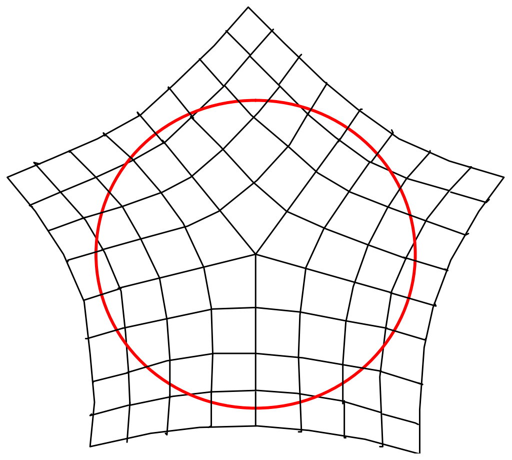
This is a circle centered at the extraordinary point in the parameter space (See Fig. 3(a)). Note that some of the parameters (u, v) in the characteristic map might be outside the unit circle [17, 18], i.e., $u^2 + v^2 > 1$ is possible. Hence the actual blending area is smaller than the whole domain. It should be pointed out that the blending area defined here is different from the one used in [18], which is defined by $u^2 + v^2 \leq \lambda_n$ with λ_n being the sub-dominant eigenvalue of the subdivision matrix corresponding to a valence n extraordinary vertex. The reason for this difference is because we want to maximize the blending areas and overlapping of blending areas does not matter in our case.

The blending weight function $W(u, v)$ must satisfy the condition $0 \leq W(u, v) \leq 1$ in the blending region and has to be at least C^2-continuous everywhere. We follow Levin's approach [18] to define $W(u, v)$, i.e.,

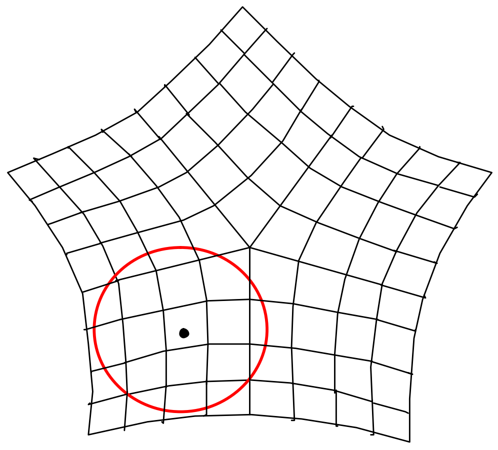
$$W(u, v) = (u^2 + v^2)(3(u^2 + v^2) - 8\sqrt{(u^2 + v^2)} + 6)$$

It is easy to see that $W(u, v)$ satisfies $0 \leq W(u, v) \leq 1$ in the region $u^2 + v^2 \leq 1$ and is C^2-continuous everywhere. At the extraordinary point, $W(u, v)$ approaches zero at the rate of $u^2 + v^2$. When near the boundary of the blending region $u^2 + v^2 = 1$, $W(u, v)$ approaches 1, with zero partial derivatives up to order 2. Hence, the resulting surface is guaranteed to interpolate the given mesh and, meanwhile, cancels out the irregularity and discontinuity of the blending surface T.

6. Blending around an arbitrarily selected point: Because we allow local adjustment of the interpolating surface, there should be a way for the system to perform blending process in regions around arbitrarily selected points of the surface, not only the images of the control vertices P_i. With a local parametrization, such a task is actually relatively easy to achieve. For example, in Fig. 3(b), to adjust the interpolating surface in a small area around the selected point (the one marked with a black solid circle), we first find the parameter (u_0, v_0) of the selected 3D point in the parameter space of the local parametrization that covers the selected point, and then find the biggest circle in the parameter space whose center is (u_0, v_0) (the red circle in Fig. 3(b)). This circle defines the blending area for the selected point.



(a) Blending around the image of an original vertex



(b) Blending around an arbitrary point

Figure 3: Parameter space for a vertex of degree 5 and for an arbitrarily selected point.

To speed up the search of (u_0, v_0), we can simply choose the closest (u_0, v_0) in the k-times refined characteristic map ($k \in Z$ and $k \geq 0$). Once we have (u_0, v_0), we still need to find the biggest radius for the blending area. Again we compare all the distances from the selected point (u_0, v_0) to all the boundary parameter values in the k-times refined characteristic map ($k \in Z$ and $k \geq 0$) and the smallest one is the radius of our blending area, denoted r_0. In addition, as mentioned above, the blending area should not include the parameter point (0,0). So we also need to compare r_0 with the distance between $(0, 0)$ and (u_0, v_0) and the smaller one is called r. Therefore the blending area for the selected point can be defined as follows.

$$(u - u_0)^2 + (v - v_0)^2 < r^2$$

The corresponding blending weight function $W(u, v)$ is defined by the following quartic formula [18].

$$W(u, v) = \rho^2(3\rho^2 - 8\rho + 6),$$

where

$$\rho = \frac{\sqrt{(u - u_0)^2 + (v - v_0)^2}}{r}.$$

It is easy to see $\rho \leq 1$ and $W(u, v)$ satisfies $0 \leq W(u, v) \leq 1$ in the blending region and is C^2-continuous everywhere. Note that at the selected point, $W(u, v)$ approaches zero. When near the boundary of the blending region, $W(u, v)$ approaches 1, with zero partial derivatives up to order 2. Consequently, it can still cancel out the irregularity and discontinuity of the blending surface T while locally modify the shape of the interpolating surface according to the need of the user.

6.1. Revisit Construction of Blending Surface T: In the above section, we have discussed how to construct an initial blending surface T_0 around vertices to be interpolated. In this section, we show how to construct a blending surface T_i around an arbitrarily selected point. T_i, like T, should also be easy and efficient to construct. Again, we can use affine transformation to construct T_i from S_i. The scaling factor components of the affine transformation matrix are easy to determine, simply compare the dimensions of T_i and S_i. The question is how to determine the offset components of the affine transformation matrix. Note that, unlike the case of T_0 where the offset components of the affine transformation matrix are determined by the vertex to be interpolated and its limit point on S, in this case, there is no point in the given mesh that corresponds to the selected point on S_i. In an interactive environment, such a point can be specified by the user. But how should such a point and, consequently, the offset vector be determined for an automatic system?

We propose to determine the offset vector for each selected 3D point by constructing a Hermite surface for the patch that covers the selected point. For example, if the offset vectors for the four vertices of the patch are D_1, D_2, D_3 and D_4, then we construct a Hermite surface patch $H(u, v)$ based on D_1, D_2, D_3 and D_4. The tangent vectors at the four corners required for the construction of H are set to the partial derivatives of the limit surface S

at the four corners. The offset vector for a selected point with parameter value (u_0, v_0) in S_i, is set to $H(u_0', v_0')$, where (u_0', v_0') can be determined by linearly mapping quadrilateral $[0, \cos(2i\pi/n)] \times [0, \sin(2i\pi/n)]$ to a unit square.

7. Interpolation of Normal Vectors: Direction of normal vectors specified at vertices of the given mesh can also be interpolated. The key is to modify the construction process of the blending surface T_0 so that it would have the same normals (actually the same partial derivatives) at the extraordinary points. This can be easily achieved by rotating each piece of T_0 with appropriate X, Y and Z rotation factors after the above mentioned translation and scaling process. This is possible because each piece of T_0 interpolates one point of P only. Hence we have a blending surface T_0 that not only interpolates the given mesh P but normals specified at some or all vertices of P as well. Because the value of $W(u, v)$ and its first partial derivatives at $(0, 0)$ are all zero, the resulting interpolating surface $\bar{S}$ then satisfies $\frac{\partial T(u,v)}{\partial u} = \frac{\partial \bar{S}(u,v)}{\partial u}$ and $\frac{\partial T(u,v)}{\partial v} = \frac{\partial \bar{S}(u,v)}{\partial v}$. In other words, $\bar{S}$ and T_0 have the same normal. Hence, with one more Affine transformation (actually they can be combined into a single matrix to save computation time), we can construct an interpolating surface that not only interpolates the given mesh, but normals at all or some of the vertices of the mesh as well.

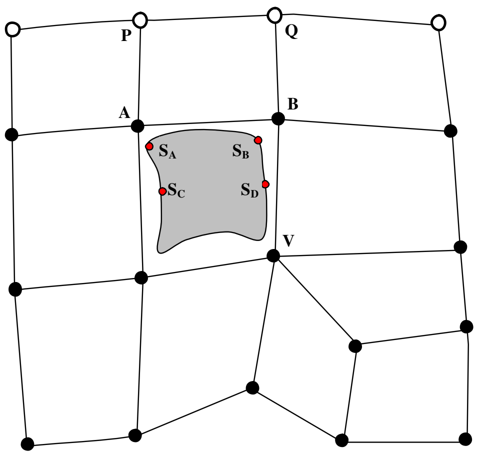
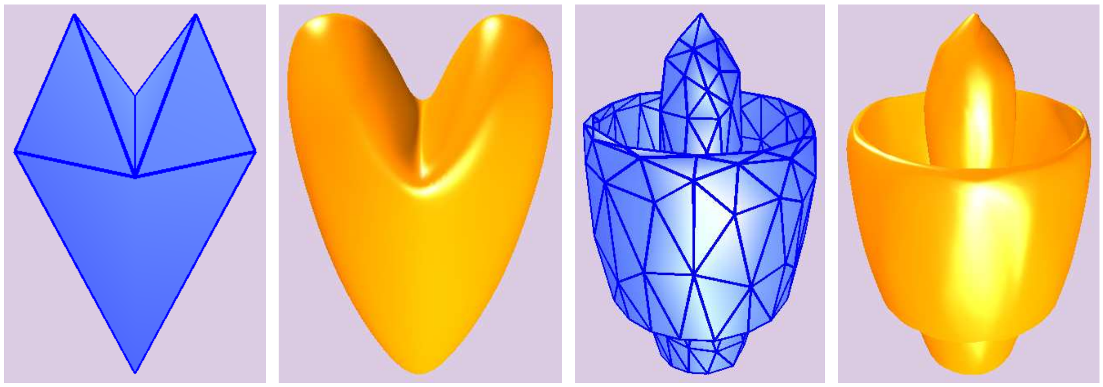


Figure 5: Handling open meshes

8. Handling Open Meshes: The interpolation process developed in the previous sections can not be used for open meshes, such as the one shown in Fig. 4(c), directly. This is because boundary vertices of an open mesh have no corresponding limit points, nor derivatives. Therefore, the Affine transformation matrix required for the construction of T_0 cannot be built directly. One way to overcome this problem is to add an additional ring of vertices along the current boundary and connect the vertices of this ring with corresponding vertices of the current boundary to form an additional ring of faces. The newly added vertices are called *dummy vertices*. We then apply the interpolation method to the extended open mesh as to a closed mesh except that there are no actions taken for the dummy vertices. This technique of extending the boundary of a given mesh is similar to a technique proposed for uniform B-spline surface representation in [1]. Note that in this case, the interpolation process is not based on the limit surface of the given mesh, but the limit surface of the extended mesh. Therefore, the shape of the interpolating surface will be affected by locations of the dummy vertices as well. Determining the location of a dummy vertex, however, is a tricky issue, the user should not be burdened by such a tricky task. In our system, this is done by using locations of the current boundary vertices of the given mesh as the locations of the dummy vertices. Note that our interpolation technique is performed directly on the limit surface, hence there is no need to care about positions of the dummy vertices after interpolation.

Another approach to handle open mesh interpolation is to modify the proposed interpolation method for closed meshes. Note that our method is locally adjustable. Hence the limit point of a vertex actually can be moved to anywhere, as long as the interpolation requirement are satisfied. Consider the mesh shown in Fig. 5 where vertices marked with circles, like P and Q, are boundary vertices and vertices marked with solid circles, such as V, A and B, are interior vertices. The shaded surface patch is the corresponding limit surface where $S(A)$ and $S(B)$ are the images of A and B, respectively, $S(C)$ and $S(D)$ are the images of the corresponding edge points, respectively. According to our interpolation method, $S(A)$ and $S(B)$ should be moved to A and B, respectively. However, to interpolate the boundary points P and Q, we can modify our approach such that $S(A)$ and $S(B)$ are moved to P and Q, respectively, and $S(C)$ and $S(D)$ are moved to A and B, respectively. The resulting surface then interpolates all the vertices of the given open mesh.

9. Test Results: The proposed techniques have been implemented in *C++* using *OpenGL* as the supporting graphics system on the Windows platform. Quite a few examples have been tested with the techniques described



(a) Given Mesh (b) Interpolation Surface (c) Given Mesh (d) Interpolation Surface

Figure 4: Examples

here. All the examples have extra-ordinary vertices. Some of the tested results are shown in Figures 1, 4 and 6. From these examples we can see smooth and visually pleasant shapes can be obtained by locally adjusting the original limit surfaces.

In our implementation, two subdivision steps are performed on the given mesh for each example before the parametrization technique [17] is applied. The evaluation of the interpolating surfaces are based on sample parameter values of the 4-times refined characteristic maps, and we find the results to be good for most cases. For bigger patches one can use more sample points because patches do not have to be sampled uniformly.

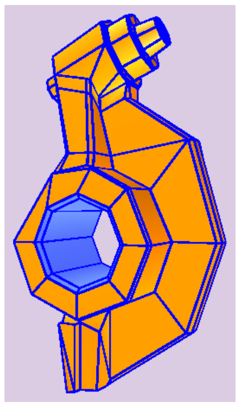
All the interpolation shown in Figures 1, 4 and 6 are done with at least two blending processes. First one is done with T_0, which is based on all the given control vertices. T_1 for the second blending process is based on all edge points of the given mesh. Some figures in the examples went through more blending processes to further improve quality of the interpolating surface. T_i's for those blending processes are selected based on, for example, face points of all patches, or parameter values $(\frac{1}{2^j}, \frac{1}{2^k})$, where j and k are integers. User interaction is also possible. For example, Figure 1(b) and Figure 1(c) both interpolate the given mesh shown in Figure 1(a), but Figure 1(c) is obtained with more local adjustment on the upper part of the teapot body. The other parts are not adjusted, hence they are exactly the same as those shown in Figure 1(b). Figure 1 shows, with user local adjustment, a better shape can be obtained after some automatic blending processes.

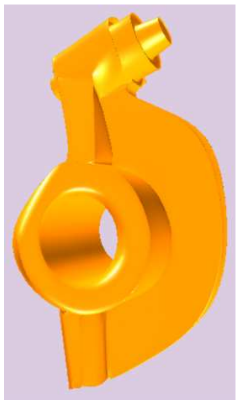
The original Utah teapot consists of four separate parts: lid, handle, body and spout. The mesh shown in Figure 1(a) is actually a set of four meshes, one for each component of the original Utah teapot. Each part is an open mesh. Although each of these meshes can be interpolated separately, Figure 1(b) and Figure 1(c) are generated by regarding them as a single mesh. The mesh shown in Figure 6(b) is another example of an open mesh with disconnected boundaries. But different from the case shown in Figure 1, which is generated by moving the vertices to some different position intentionally, Figure 4(d) is generated using additional dummy vertices in the interpolating surface construction process.

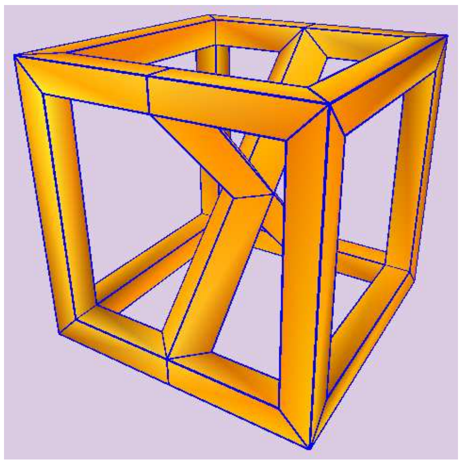
The new interpolation method can handle meshes with large number of vertices in a matter of less than a second on an ordinary PC (3.2GHz CPU, 512MB of RAM). For example, the meshes shown in Figures 1(a), 4(a), 4(c), 6(a) and 6(c) have 320, 9, 194, 354 and 66 vertices, respectively, and it takes almost no time to interpolate these relatively small meshes. Since it is a local blending process and is performed directly on the limit surface, our method can easily handle meshes with thousands of or more vertices. Hence our interpolation method is especially suitable for interactive shape design.

10. Summary: A new interpolation method for meshes with arbitrary topology is presented. The interpolation process is a local process, it does not require solving a system of linear equations. Hence, the method can handle data set of any size.

The interpolating surface is obtained by locally adjusting the limit surface of the given mesh (viewed as the control mesh of a Catmull-Clark subdivision surface) so


(a) Given Mesh


(b) Interpolating Surface


(c) Given Mesh

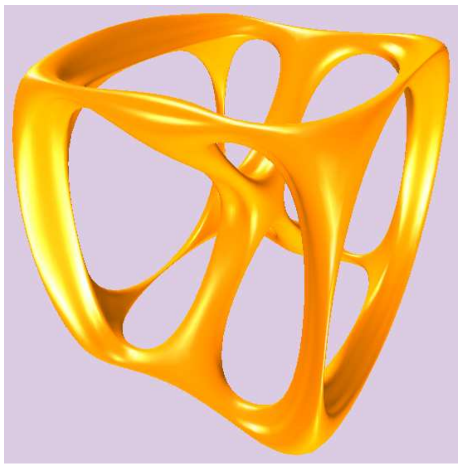

(d) Interpolating Surface

Figure 6: Examples

that the modified surface interpolates all the vertices of the given mesh. This local adjustment process can also be used to smooth out the shape of the interpolating surface. Hence, a *surface fairing* process is not needed in the new method.

The new method can handle both open and closed meshes. It can interpolate not only vertices, but normals and derivatives as well. These normals and derivative can be anywhere, not just at the vertices of the given mesh. Test results show that the new method leads to good interpolation results even for complicated data sets.

The resulting interpolating surface is not a Catmull-Clark subdivision surface. It does not even satisfy the convex hull property [18]. But the resulting interpolating surface is guaranteed to be C^2 continuous everywhere except at some extraordinary points, where it is C^1 continuous. Using a technique similar to the one presented in [18], a C^2 continuous interpolating surface can also be achieved.

11. References:

[1] Barsky B A, End conditions and boundary conditions for uniform B-spline curve and surface representation, *Computers in Industry*, 1982, 3(1/2):17-29.

[2] Catmull E, Clark J, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design*, 1978, 10(6):350-355.

[3] Doo D, Sabin M, Behavior of recursive division surfaces near extraordinary points, Computer-Aided Design, 1978, 10(6):356-360.

[4] Dyn N, Levin D, and Gregory J A, A butterfly subdivision scheme for surface interpolation with tension control, *ACM Transactions on Graphics*, 1990, 9(2):160-169.

[5] Halstead M, Kass M, DeRose T, Efficient, fair interpolation using Catmull-Clark surfaces, *ACM SIGGRAPH*, 1993:35-44.

[6] Levin A, Interpolating nets of curves by smooth subdivision surfaces, *ACM SIGGRAPH*, 1999, 57-64.

[7] Kobbelt L, Interpolatory subdivision on open quadrilateral nets with arbitrary topology, *Computer Graphics Forum*, Eurographics, V.15, 1996.

[8] Lin H, Wang G, Dong C, Constructing Iterative Non-Uniform B-spline Curve and Surface to Fit Data Points, *SCIENCE IN CHINA*, Series F, 2004, 47:315-331.

[9] Litke N, Levin A, Schröder P, Fitting subdivision surfaces, *Proceedings of the conference on Visualization* 2001:319-324.

[10] Loop C T, Smooth subdivision surface based on triangle, *Master's thesis*, Department of Mathematics, University of Utah, 1987.

[11] Maekawa T, Matsumoto Y, Namiki K, Interpolation by geometric algorithm, *Computer-Aided Design*, 2007, 39:313-323.

[12] Nasri A H, Surface interpolation on irregular networks with normal conditions, *Computer Aided Geometric Design*, 1991, 8:89-96.

[13] Nasri A H, Sabin M A, Taxonomy of interpolation constraints on recursive subdivision curves, *The Visual Computer*, 2002, 18(4):259-272.

[14] Schaefer S, Warren J, A Factored Interpolatory Subdivision Scheme for Quadrilateral Surfaces, *Curves and Surface Fitting*, 2002, 373-382.

[15] Stam J, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, *Proceedings of SIGGRAPH* 1998:395-404.

[16] Stam J, Evaluation of Loop Subdivision Surfaces, *SIGGRAPH'99 Course Notes*, 1999.

[17] Reif, U. A unified approach to subdivision algorithms near extraordinary points. *Computer Aided Geometric Design* 12, 2, 153C174, 1995.

[18] Adi Levin, Modified Subdivision Surfaces with Continuous Curvature, *Proceedings of SIGGRAPH*, 1035-1040, 2006.

[19] D. Zorin, P. Schröder, W. Sweldens, Interpolating Subdivision for meshes with arbitrary topology, *ACM SIGGRAPH*, 1996:189-192.

[20] Kestutis Karciauskas and Jörg Peters, Guided Subdivision, http://www.cise.ufl.edu/research/ SurfLab/papers/05guiSub.pdf, 2005.

[21] Warren, J., and Weimer, H., *Subdivision Methods for Geometric Design*, Morgan Kaufmann, 2002.