Free-Form Deformation with Automatically Generated Multiresolution Lattices

Yutaka Ono* Bing-Yu Chen* Tomoyuki Nishita* Jieqing Feng†
*The University of Tokyo
†Zhejang University
{yutaka-o, robin, nis}@is.s.u-tokyo.ac.jp
jqfeng@cad.zju.edu.cn

Abstract

Developing intuitive and efficient methods for shape editing is one of the most important areas in computer graphics, and free-form deformation (FFD), which is one of such methods, allows the user to deform a model easily by moving a set of control points, collectively called the lattice. Although the FFD method can be used for both global and local deformations, the user must define a suitable lattice manually or use a simple shaped lattice such as a parallelepiped. Therefore, we propose a new FFD method that automatically generates the lattices with which both types of deformations can be achieved.

Our method refines a bounding box of the model and generates a set of finer lattices, which hierarchically approximate the shape of the model. Through adjusting the control points of the generated lattices, both global and local deformations of the model can be achieved easily. Moreover, the method allows hierarchical deformation of the model by combining different levels of lattice.

1. Introduction

Recently, complex and detailed 3D models have become widely used in many fields such as the movies, computer games, and so on. Such elaborate models fascinate a lot of people, so that not only professional users but also amateur hobbyists are interested in making such models. Hence, useful methods for 3D shape design, modification, and animation have become more and more important in computer graphics.

FFD [10] is one of the methods used for editing such models. It achieves a smooth deformation of the models in a model independent way; i.e. the user can deform a model without any knowledge of its mathematical background. Hence, it is highly popular among both professional and amateur users.

The procedures of the FFD method are as follows: the user defines a deformable region of space by placing a set of control points, which together form a lattice. A model is then embedded in the deformable region. The user moves one or some of the control points to deform the deformable space and this is automatically passed onto the model.

The FFD method allows the user to deform the model intuitively and efficiently, if the lattice is properly designed. For example, the user can do global deformation such as bending, twisting, or tapering if the lattice is comprised of a few control points and the deformable space is large enough to contain all the vertices of the model to be deformed. On the other hand, he or she can do local deformation such as adding bumps if the lattice has a suitable density of the control points close to the region that is to be deformed. However, it is not easy to design suitable lattices and to capture the correspondence between lattices and the model to be deformed since the original FFD method allows only parallelepiped lattices.

To solve this problem, researchers have introduced methods to generate more general lattice structures. Although these methods allow a greater inventory of deformable space, some of them, e.g. extended free-form deformation (EFFD) [2], restrict the flexibility of the original method. MacCracken and Joy's method [7] allows lattices to have arbitrary topology, however, this cannot be applied to lattices with many control points since the computational cost is relatively high. Moreover, almost all of these methods concentrate only on deformation techniques with the available lattices, none of them have introduced a way to automatically generate appropriate lattices. Thus, the user must carry out the tedious task of defining lattices manually, and this obviously decreases his or her productivity.

Other approaches for efficient and intuitive deformation are introduced by using an axis [6] or some wires [11] instead of a lattice. Although these methods provide a simpler user-interface than traditional ones, they also restrict the flexibility, and again, the user must manually define a suitable axis or wires.

Therefore, in this paper, we propose a method to help users deform 3D models by generating FFD lattices automatically while keeping the flexibility of the original FFD method. In our method a bounding box for the model is generated first, and this is also identified with the lattice. Then, the lattice is refined hierarchically and a set of lattices called *multiresolution lattices* is generated, so that the user can select from them an appropriate lattice according to his or her purpose, i.e. a lattice with a few control points for global deformation or a lattice closely approximating the shape of the model for local deformation. Moreover, by refining not only the unmodified lattices but

also the modified ones, it is possible to do *hierarchical deformation*, with which the user can deform the model hierarchically without redefining the lattices from the bounding box. Thus, our approach liberates the user from defining lattices manually, strengthens the intuition of FFD, and increases the user's productivity.

2. Automatic generation of multiresolution lattices

Before describing the details of our method, we define the terms used throughout this paper by reference to [7]:

- A *lattice* is defined as a set of control points and an associated set of pairs that specifies the connectivity of the control points.
- An *edge* of the lattice is defined by two control points that are connected in the lattice.
- A *face* of the lattice is defined by a minimal connected loop of control points.
- A *cell* of the lattice is the region of space bounded by a closed set of faces.

Our deformation approach allows lattices with the following properties, called *valid lattices*:

- The lattice is well-connected.
- All cells of the lattice are closed, not containing any holes.
- The lattice is not self-intersecting.

2.1. Octree subdivision lattices

To do the deformation, the region of the model to be deformed is first determined by the user. Although the region maybe cover the whole model or just only some parts of it, our approach can be applied to both types.

Given the region to be deformed, our method makes the lattices hierarchically approximate to the shape of the region by repeating 3D subdivisions of the bounding box, which is identified with the lowest-level lattice of the multiresolution lattices. The lowest-level lattice could be an axis-aligned bounding box (AAB), an oriented bounding box (OBB) [3], or a minimum-volume bounding box. In most cases, we found that the OBB was the most intuitive lattice for the user since its axes follow the shape of the principal components of the region to be deformed. After defining the lowest-level lattice, we generate multiresolution lattices using *octree subdivision rules* that are similar to the rules for octree subdivision of 3D space. The lattices produced by applying these rules consist of the following types of new control point:

- A *cell control point* is that defined by the average of the control points of the lattice which define the cell.
- A *face control point* is that defined by the average of the control points of the lattice which define the face.
- An *edge control point* is the midpoint of the edge.

At each subdivision step, the above three types of control point are generated for each cell, face and edge of the lattice. These new control points and old control points are reconnected to create a new lattice according to the following reconnection rules: (1) each new cell control point is connected to the new face control points generated from the faces that define the old cell; (2) each new face control point is connected to the new edge control points generated from the edges that define the old face; (3) each new edge control point is connected to the two old control points that define the old edge.

Using the octree subdivision rules, we generate *octree subdivision lattices* as follows:

1. Define the lowest-level lattice.
2. Create a finer lattice by applying the octree subdivision rules to the current-level lattice.
3. Cut the cells of the new lattice that do not contain any vertex of the model.
4. Repeat steps 2 and 3 until the lattice has user specified details.

Since the lattices, which are generated by the above procedure, can approximate the shape of the model hierarchically and are comprised of uniformly-located control points as shown in Figure 1, the user can do any level of deformation with ease and intuition. Moreover, since octree subdivision rules can be applied to any valid lattice, this method can be applied not only to unmodified lattices but also to the user-modified ones. This allows the user to do *hierarchical deformation* (see Section 4).

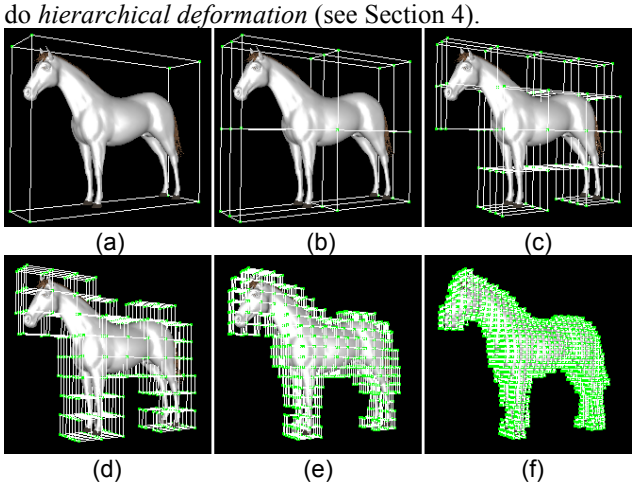
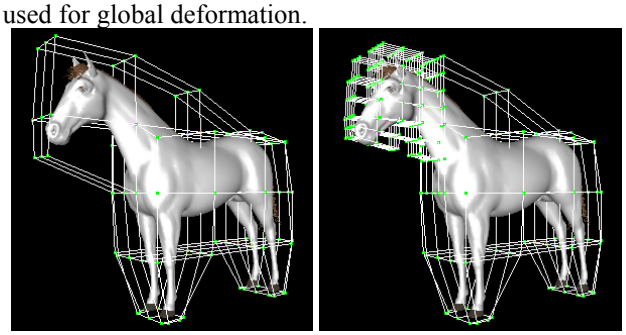


Figure 1. Octree subdivision lattices generated for a whole horse model. (a) The lowest-level lattice. (b) through (f) Finer lattices.

2.2. Extensions of octree subdivision lattices

Although the octree subdivision lattices can approximate the shape of the model, they sometimes fail to do a sufficiently good approximation around its boundary, especially when the level of the lattice is low and each cell of the lattice is relatively large compared to the space occupied by the vertices of the model. Since it is desirable that the lattices approximate the model as closely as possible for intuitive deformation, we developed a method that improves the octree subdivision lattices. In this method, the control points on the boundary of each axis

are moved close to the model along the axis while maintaining a certain offset. We call the lattices generated by this method *shrink octree subdivision lattices* as shown in Figure 2 (a). These lattices are useful when the user deforms a model globally with low-level lattices that consist of a few cells. However, applying this method to a high-level lattice often generates a lattice with scattered-looking control points if we have no proper offset. Therefore shrink octree subdivision lattices should normally be used for global deformation.



(a) (b)

Figure 2. (a) A shrink octree subdivision lattice and (b) a local octree subdivision lattice corresponding to Figure 1 (c).

Although the user can do any level of deformation by selecting a proper lattice among multiresolution lattices, to make the user to focus on the regions of the model which he or she wants to deform locally, we also allow the user to apply octree subdivision rules only to user-specified cells. This method is similar to the local subdivision proposed in [8]. Although it complicates the local topological structure, the resulting lattice is still valid as shown in Figure 3. This method adds much flexibility to the lattices, and allows the user to deform some regions of the model locally and the rest globally without redefining the lattices as shown in Figure 2 (b).

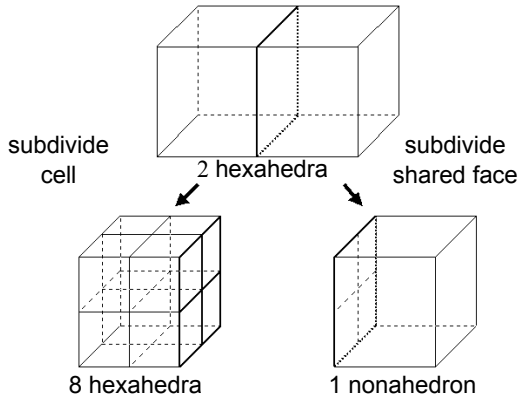


Figure 3. A local subdivision of a hexahedron.

3. Deformation process using one lattice

To deform a model, we parameterize each vertex of the model within the deformable space defined by the lattice using the Catmull-Clark subdivision volumes introduced by MacCracken and Joy [7]. It should be understood that we can parameterize each vertex more easily by regarding the lattice as b-spline volume when the lattice is comprised of only hexahedral cells, since the b-spline subdivision method is a special case of the Catmull-Clark subdivision method. However, if the lattice is locally subdivided, it cannot be regarded as b-spline volume. This is the reason why we use the Catmull-Clark subdivision. The Catmull-Clark subdivision method is applied to the user-editing lattice, and then produces a sequence of *subdivided lattices* that successively approximate the deformable space. Each vertex of the model is then parameterized with the minimal cell of the subdivided lattices that contains the vertex. NOTICE: the subdivided lattices are different from the multiresolution lattices described in the previous section. The former ones are used only for parameterization, while the latter ones are actually displayed and modified by the user.

3.1. Property of Catmull-Clark subdivision volumes

The Catmull-Clark subdivision rules for producing subdivided lattices are described in Appendix. These subdivided lattices have some special features, which are used in our deformation process. One of the features is concerned with the connectivity of each control point. To describe them, we define the *valence* of a point within the cell of a lattice to be the number of the cell's edges that connect the point. Given the cell of a lattice, the subdivision rules create a new cell at each point of the cell. The new cell generated at a point of valence v has $2v$ 4-sided faces, 2 points of valence v, and $2v$ points of valence 3. For example, a new cell generated at a point of valence 3 has 6 4-sided faces and 8 points of valence 3, that is, a hexahedron. Additionally, we define a *normal point* to be a point whose valence is equal to 3 and an *extraordinary point* whose valence is not equal to 3. In the case of simple octree subdivision lattices (not locally subdivided), it can be easily proved that the Catmull-Clark subdivision rules produce subdivided lattices comprised of only normal points and hexahedral cells, and the number of cells multiplies by a factor of eight at each subdivision. Extraordinary points appear only when some of the cells are locally subdivided. Thus, the size of required memory space increases by a factor of about eight at each subdivision.

Another feature is concerned with a local property of the subdivided lattices. As in the case of the Catmull-Clark subdivision surfaces, each control point of the Catmull-Clark subdivision volumes influences only limited regions of the volumes. This is explained by using the symmetric bivariate function *dist*, which is recursively defined by two control points:

- $dist(p, p) = 0$, where p is a control point of the lattice.
- $dist(p, q) = 1$, where p and q are different control points contained in the same cell.
- $dist(p, q) = i + 1$, where p and q are different control

points, $q \notin \{r \mid dist(p,r) \le i\}$, and there exists a control point s such that $dist(p,s) = i$ and $dist(s,q) = 1$.

Simple observation of the subdivision rules in Appendix shows that at each subdivision step, one control point p does not influence the positions of the newly generated points generated for the cells, faces, edges, and points that contain at least one point outside of $D_{p,1} = \{q \mid dist(p,q) = 1\}$. This property is used in our deformation algorithm described in the next section.

3.2. Deformation algorithm

Using the features of the Catmull-Clark subdivision volumes, we improved the deformation process outlined in [7]. The essence of deformation process is basically the same as the original FFD method in [10]. First, a lattice is defined by the user using our lattice generation method. Next, each vertex of the region of the model to be deformed is parameterized by using a sequence of the Catmull-Clark subdivided lattices: each vertex is "tagged" with a pointer to the minimal cell of the subdivided lattices that contains the vertex and its local coordinate in the cell. Then, the lattice is modified by the user. Finally, each vertex of the model is relocated by using its local coordinate in the minimal cell and a sequence of the subdivided lattices that are generated from the modified lattice. However, there is a problem in the parameterization and relocation steps: to execute the Catmull-Clark subdivision to a lattice, we must keep much of its topological information, which increases by a factor of eight with each subdivision step, thus intensively increasing the required memory space. Therefore, it is almost impossible to subdivide high-level octree subdivision lattices such as shown in Figure 1 (e) or (f) comprised of a great many cells used for local deformation.

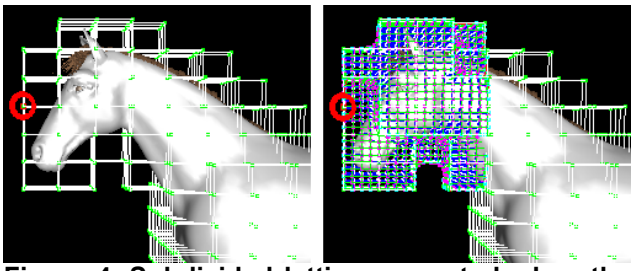


Figure 4. Subdivided lattice generated when the user moves the circled control point.

Fortunately, if the user moves one control point p of a lattice, only the vertices of the model that are enclosed by the newly generated points for $N_{p,2} = D_{p,0} \cup D_{p,1} \cup D_{p,2}$ may be changed their positions since p does not influence the positions of newly generated points for the boundary of $N_{p,2}$. Moreover, the newly generated points for the boundary of $N_{p,2}$ may be located outside of $N_{p,2}$, because they are influenced by $N_{p,3}$. Therefore, the candidate vertices of the model for relocation are all contained in the cells defined by $N_{p,3}$. Finally, since the newly generated points for $N_{p,3}$ are influenced by $N_{p,4}$, we also have to take $D_{p,4}$ into consideration. In summary, when one control point p of the lattice is moved by the user, it is sufficient to subdivide the cells consisted by $N_{P,4}$ as shown in Figure 4, then parameterize and relocate the vertices inside the cells consisted by $N_{P,3}$.

Taking this into account, we define our deformation algorithm that use only one of the multiresolution lattices as follows:

1. Multiresolution lattices are generated for the model to be deformed.
2. One of the multiresolution lattices is selected by the user.
3. The given model and the selected lattice are duplicated to be two pairs *A* and *B*. Both of them contain a lattice and a model. In the following steps, a lattice L_A and a model M_A of the one pair *A* are displayed and deformed by the user, while the other lattice L_B and the other model M_B of the other pair *B* are used only for parameterization so that it is never displayed nor deformed.[1]
4. A set of control points S_{P_A} of the lattice L_A is specified by the user as the moving points set.
5. A set of control points S_{P_B} of the lattice L_B that corresponds to S_{P_A} of L_A is specified.
6. A set of subdivided lattices $S_{L_B} = \{L_{B_1}, L_{B_2}, ..., L_{B_n}\}$ is defined, where L_{B_i} is a subdivided lattice generated by applying the Catmull-Clark subdivision rules *i* times to $N_{S_{P_B},4} = \bigcup_{q \in S_{P_B}} N_{q,4}$.
7. Each vertex of the model M_B contained in $N_{S_{P_B},3}$ is parameterized with the minimal cell of the lattices in S_{L_B} that contains the vertex.
8. The lattice L_A is modified by the user by moving the points in S_{P_A}.
9. S_{L_B} is recalculated by using the corresponding control points $N_{S_{P_A},4}$.
10. Each vertex of the model M_A contained in $N_{S_{P_A},3}$ is relocated by using the parameters of the corresponding vertex of M_B calculated at step 7.
11. Repeat steps 4 through 10 until the user finishes deformation.

As in the algorithm in [7], the above algorithm cannot do exact deformation since the Catmull-Clark subdivisions that approximate the deformable space are done only a finite number *n* times, which is described at step 6, and the deformation results are improved by increasing *n*. The most important point in our algorithm is that only a limited region of the lattice is subdivided, so that it can be applied to deformation with lattices of a great many cells. On the other hand, if the number of control points of the lattice is small enough, we subdivide the whole lattice and parameterize all the vertices of the model only once at the preprocess step. We need neither to subdivide at each change of the moving points set nor to store unmodified lattice and model.

[1] Both L_B and M_B are necessary because appropriate parameterization cannot be obtained with a deformed lattice that may not be valid.

The local coordinate of each vertex in a cell is calculated in two ways as in [7]. For a hexahedral cell, it is calculated by trilinear interpolation. For other types of cell, we partition the cells into tetrahedra, and then calculate the weights of the four points of the tetrahedron that include the vertex.

4. Deformation process using multiresolution lattices

The process for hierarchical deformation using more than one lattice of the multiresolution lattices is slightly different from the process of just using one lattice, which is described in the previous section, since the user could change the deformation levels from global to local and vice versa.

4.1. From global to local deformation

Hierarchical deformation from global to local proceeds as follows:

1. The user globally deforms the model with a low-level lattice.
2. A finer lattice is generated by applying the octree subdivision rules to the modified lattice.
3. The user locally deforms the model with the finer lattice.
4. Repeat steps 2 and 3 if a finer deformation is needed.

To extend the algorithm in Section 3 to achieve this kind of hierarchical deformation is rather simple, but there is one problem caused by the difference between the octree subdivision and the Catmull-Clark subdivision rules.

Let O and C be the maps from low-level lattice to high-level one defined by the octree and the Catmull-Clark subdivision rules respectively, L_0 be the lowest-level lattice, and L_i be $O^i(L_0)$, where i is the level of the multiresolution lattices. To deform the model by using L_i, we first calculate a set of parameters P_i of vertex set V using $C^n(L_i)$, which is generated by applying the Catmull-Clark subdivision rules n times to L_i. After L_i is modified by the user, L_i is become L_i', and each vertex in V are relocated by using $P_i' = P_i$. If we apply the octree subdivision rules to L_i' now to get $L_{i+1}' = O(L_i')$ for local deformation, it is needed to recalculate the parameters P_{i+1}' from L_{i+1}'. Unfortunately, in many cases, it is impossible to get P_{i+1}' directly because L_{i+1}' may not be valid, so we use P_{i+1} calculated from $L_{i+1} = O(L_i)$ instead. But it is easy to find $P_{i+1} \neq P_{i+1}'$ in general, hence the vertex position calculated by P_{i+1} is slightly different from the original one. This is caused by the difference between O and C.

Although we can solve this problem by using the Catmull-Clark subdivision rules instead of the octree subdivision ones when we generate multiresolution lattices, the lattices generated by using the Catmull-Clark subdivision rules sometimes produce rather scattered control points and not easy to deform. Moreover, it is almost impossible to create high-level lattices because of its computational cost. Therefore, we make a lookup table here to correct the gap.

4.2. From local to global deformation

Hierarchical deformation from local to global is rather difficult because the low-level lattices regenerated by applying the inverse operation of octree subdivision rules to the high-level lattice may not contain whole the region to be deformed. Therefore, we provide this kind of deformation by defining a new low-level lattice that encloses both the model and the high-level lattice. The control points of the high-level lattice are also deformed by the modification of the new low-level lattice as the model. Therefore, the user can do the global deformation even if some local deformations are performed and vice versa.

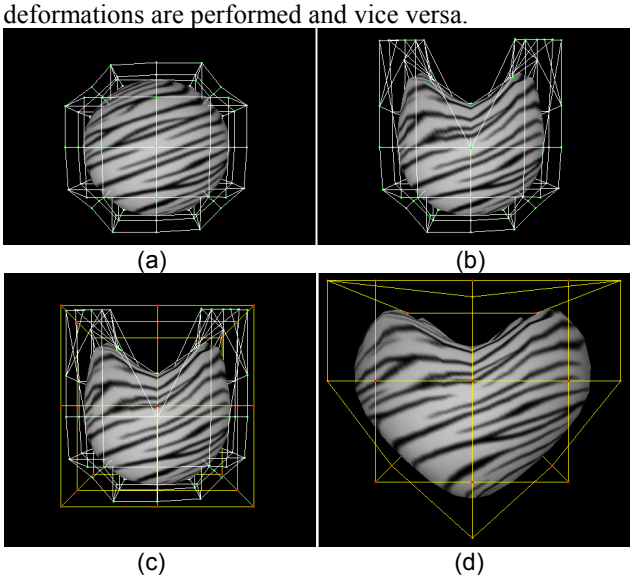


Figure 5. Hierarchical deformation from local to global. (a) A model with a high-level lattice. (b) Local deformation is performed to the model. (c) A low-level lattice is generated to enclose the high-level lattice and the deformed model. (d) Global deformation is performed to the model through the low-level lattice. (The deformed high-level lattice is not shown.)

Figure 5 shows an example of hierarchical deformation from local to global. First, a sphere model in Figure 5 (a) is performed by a local deformation with a high-level shrink octree subdivision lattice as shown in Figure 5 (b). Next, a new low-level lattice in Figure 5 (c) that encloses both the high-level lattice and the deformed model is defined for doing global deformation. And then, the model is globally deformed as shown in Figure 5 (d). Since the high-level lattice is also deformed by the low-level lattice, it is possible to go back for the local deformation with the deformed high-level lattice.

5. Results

Figure 6 shows a global deformation using a shrink octree subdivision lattice. The user can easily do such global deformation by using shrink octree subdivision lattices. Figure 7 shows a process of hierarchical deformation. The model is first subjected to global deformation, and then successively to local deformations without redefining the lattices from the bounding box. Since refined lattices generated from a deformed lattice keep track of the features of the deformed model, the user can intuitively continue to deform the model, which is one of the main advantages of our method.

We evaluated the performance of our FFD system by calculating the required memory space and frame rates on a PC with an Intel Pentium4 1.7GHz, 256MB memory, and nVIDIA GeForce3 graphics accelerator. The results are shown in Table 1 and Table 2. The data structure used to keep topological information was like the loop edge data structure (LEDS) [9], a variant of the radial-edge data structure [12][13]. Table 1 shows that more than 20MB are required to do the Catmull-Clark subdivision twice to $N_{p,3}$ since it produces at least $32\times32\times32$ cells. Frame rates shown in Table 2 are estimated while the user moves one control point of each octree subdivision lattice to deform three polygonal models of different complexity. The Catmull-Clark subdivision is applied to each lattice twice during deformation. The most time consuming steps in the deformation algorithm are the parameterization and relocation steps. On the other hand, the time to create multiresolution lattices is almost negligible.

Table 1: Required memory sizes of our system to keep topological information of the Catmull-Clark subdivision volumes.

#Catmull-Clark subdivisions to a cube	#cells	required memory (byte)
0	$1\times1\times1$	1,296
1	$2\times2\times2$	7,376
2	$4\times4\times4$	48,336
3	$8\times8\times8$	345,488
4	$16\times16\times16$	2,601,744
5	$32\times32\times32$	20,170,256

Table 2: Frame rates estimated with different levels of lattice and different complexities of model.

#octree subdivisions to a bounding box	frame rate (fps)		
	model with 495 vertices	model with 22,886 vertices	model with 53,417 vertices
0	50.0	4.54	3.69
1	25.4	3.57	2.73
2	5.12	2.54	2.19
3	3.13	1.97	1.80
4	4.44	2.56	2.30
5	4.16	2.02	1.90

These data show that our system is still memory intensive, and may be slower than those using other simpler FFD methods. However, our system can still allow the user to deform models interactively even at these frame rates and cut through the lattice definition time from the user. One of the main features of our method is that after three steps of octree subdivision of the bounding box, the frame rate for each model's deformation doesn't decrease very much or even increases for further octree subdivisions. This is because we apply the Catmull-Clark subdivision only to the limited region of the lattice, and after each step of octree subdivision, each cell becomes smaller and contains less vertices of the model. For these reasons, this method can be applied to even more complex lattices while the computation time doesn't depend much on their complexity.

6. Conclusion and future work

In this paper, we have proposed an automatic lattice generation method that allows the user to do hierarchical deformation and a new FFD method that allows the lattices to have much complexity by subdividing only a limited region of the lattices. Our implemented system is designed to be easy-to-use for all users, whether beginners or well-trained. For example, when beginners use our system to deform a model, they just have to press buttons until the desired details are added to the lattice and to move some of its control points.

As the example shown in Figure 7, the user can easily do the model deformation that needs both global and local deformations with our method since multiresolution lattices keep track of the features of the deformed model at each level of deformation. However, if the user does deformation with other methods, he or she needs to define a proper lattice at every level of deformation manually, and it is a very time-consuming task to make the lattice fit to the model, which our multiresolution lattices naturally achieve.

However, although we only subdivide limited regions of lattices, there is still a memory limitation, which allows only a few subdivision steps. We can partly overcome this problem by adaptively using b-spline parameterization and Catmull-Clark subdivision. Moreover, to combine our method with direct manipulation methods such as [4] and [5] will makes our FFD system much more useful. We shall be developing these and creation of more easy-to-use lattices in our future work.

Acknowledgement

One of the authors, Jieqing Feng, is partially supported by National Natural Science Foundation of China (No.69903008)

References

[1] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes", *Computer-Aided Design*, Vol. 10, 1978, pp. 350-355.

[2] S. Coquillart, "Extended free-form deformation: a sculptur-

ing tool for 3D geometric modeling", *ACM Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, Vol. 24, No. 4, 1990, pp. 187-196.
[3] S. Gottschalk, M. C. Lin, and D. Manocha, "OBB-Tree: a hierarchical structure for rapid interference detection", *ACM SIGGRAPH 96 Conference Proceedings*, 1996, pp. 171-180.
[4] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations", *ACM Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, Vol. 26, No. 2, 1992, pp. 177-184.
[5] S. Hu, H. Zhang, C. Tai, and J. Sun, "Direct manipulation of FFD: efficient explicit solutions and decomposible multiple point constraints", *The Visual Computers*, Vol. 17, No. 6, 2001, pp. 370-379.
[6] F. Lazarus, S. Coquillart, and P. Jancene, "Axial deformations: an intuitive deformation technique", *Computer-Aided Design*, Vol. 26, No. 8, 1994, pp. 607-613.
[7] R. MacCracken and K. I. Joy, "Free-form deformations with lattices of arbitrary topology", *ACM SIGGRAPH 96 Conference Proceedings*, 1996, pp. 181-188.
[8] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk, "Virtual clay: a real-time sculpting system with haptic toolkits", In *ACM Symposium on Interactive 3D Graphics Proceedings*, 2001, pp. 179-190.
[9] S. A. McMains, "Geometric algorithms and data representation for solid freeform fabrication", PhD thesis, Department of Computer Science, University of California, Berkeley, 2000.
[10] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models", *ACM Computer Graphics (SIGGRAPH 86 Conference Proceedings)*, Vol. 20, No. 4, 1986, pp. 151-160.
[11] K. Singh and E. Fiume, "Wires: a geometric deformation technique", *ACM SIGGRAPH 98 Conference Proceedings*, 1998, pp. 405-414.
[12] K. Weiler, "Topological structures for geometric modeling", PhD thesis, Rensselaer Polytechnic Institute, 1986.
[13] K. Weiler, "The radial edge structure: a topological representation for non-manifold geometric boundary representations", *Geometric Modeling for CAD Application (First IFIP WG5.2 Working Conference Rensselaerville),* 1998, pp. 3-36.
[14] D. Zorin, P. Schröder, T. DeRose, J. Stam, and J. Warren, "Subdivision for modeling and animation", *ACM SIGGRAPH 99 Conference Course Notes*, No. 37, 1999.

Appendix: Subdivision rules of Catmull-Clark volumes

The Catmull-Clark subdivision volumes method is an extension of the Catmull-Clark subdivision surfaces method [1], which successively refines the surface occupied by a set of control points. Analogously, the subdivision algorithm for volumes successively refines the 3D space occupied by a lattice. To achieve finer representations of the original lattice, the subdivision algorithm produces four types of new point:

- A *cell point* is that defined by the average of the control points of the lattice which define the cell.
- A *face point* is that defined by the weighted average: $F = \frac{C_0 + 2A + C_1}{4}$, where C_0 and C_1 are the cell points of the two cells on either side of the face and A is the face centroid.
- An *edge point* is that defined by the weighted average: $E = \frac{C_{avg} + 2A_{avg} + (n-3)M}{n}$, where C_{avg} is the point defined by the average of the cell points of the cells that contain the edge, A_{avg} is the point defined by the average of the face centroids of the faces that contain the edge, M is the midpoint of the edge, and n is the number of faces that contain the edge.
- A *vertex point* is that defined by the weighted average: $V = \frac{C_{avg} + 3A_{avg} + 3M_{avg} + P}{8}$, where C_{avg} is the point defined by the average of the cell points of the cells that contain the control point, A_{avg} is the point defined by the average of the face centroids of the faces that contain the control point, M_{avg} is the point defined by the average of the midpoints of the edges that radiate from the control point, and P is the control point itself.

At each subdivision step, cell points, face points, and edge points are generated for all cells, faces, and edges of the lattice respectively and old control points are replaced by new vertex points. These new control points are reconnected to create a new subdivided lattice according to the following reconnection rules: (1) each new cell point is connected to the new face points generated for the faces that define the old cell; (2) each new face point is connected to the new edge points generated for the edges that define the old face; (3) each new edge point is connected to the two new vertex points that are generated for the control points which define the old edge. NOTICE: the new control points are only used to generate finer Catmull-Clark volumes and not replace the original control points for user's controlling.

The control points on the boundary of the lattice are generated in particular ways to prevent resulting space from shrinking:

- A c*orner point* is that contained in only one cell of the lattice. In the refinement process, the position of a corner point remains unchanged.
- A s*harp edge* is that contained in only one cell of the lattice. The edge point of a sharp edge is the midpoint of the edge.
- A s*harp point* is that joining two sharp edges. The vertex point of a sharp point is defined by the weighted average: $V = \frac{M_1' + 2P + M_2'}{4}$, where M_1' and M_2' are the

midpoints of two sharp edges that contain the control point, and P is the control point itself.

- All other face, edge, and vertex points on the boundary are generated according to the Catmull-Clark rules for surfaces [1].

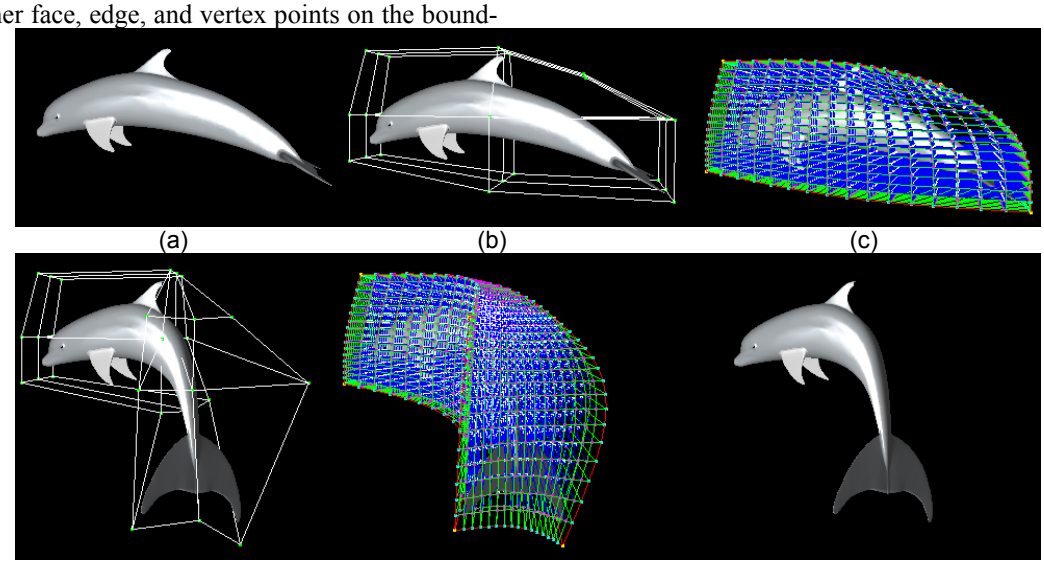


Figure 6. A global deformation of a dolphin model by using a low-level shrink octree subdivision lattice. The colored regions in (c) and (e) represent the Catmull-Clark subdivided lattice.

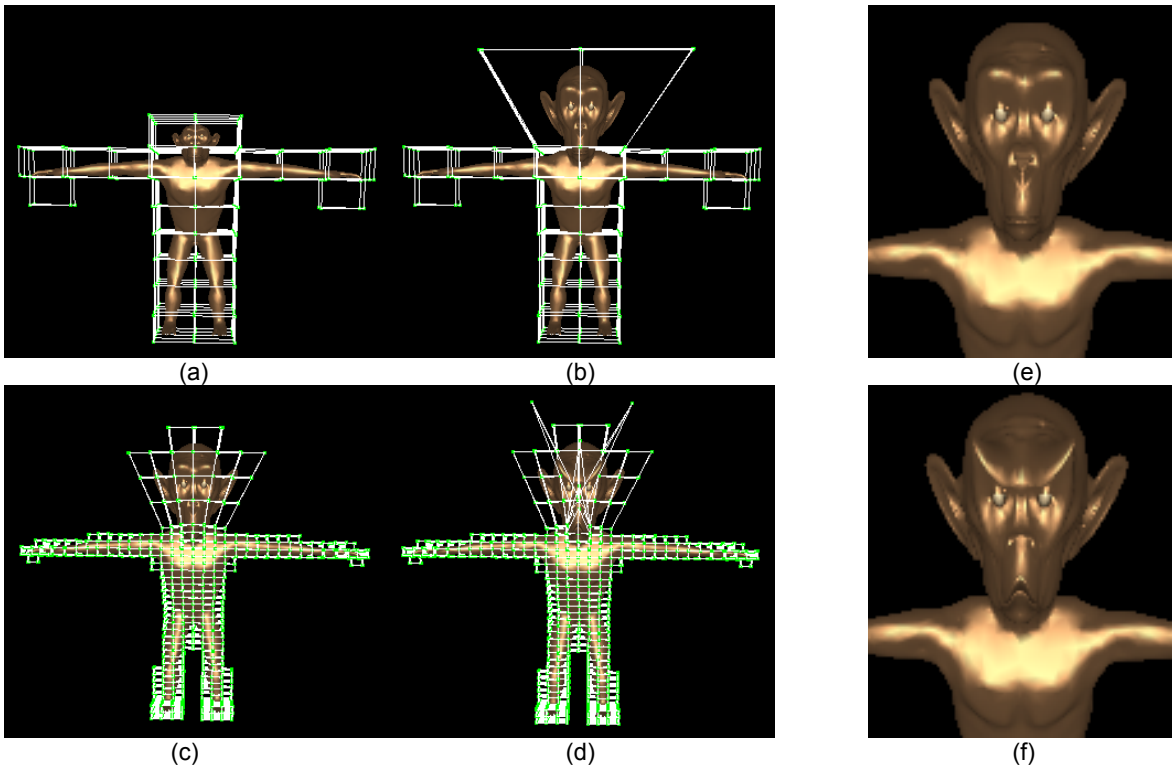


Figure 7. Hierarchical deformation of a chimpanzee model. (a) Original model with a low-level lattice. (b) Deformed model with deformed low-level lattice. (c) Deformed model with a high-level lattice generated from that of (b). (d) Further deformed model with deformed high-level lattice. (e) Closed up view of (b) and (c). (f) Closed up view of (d).