The Visual Computer

Free-form deformation with weighted T-spline

Wenhao Song,
Xunnian Yang

Department of Mathematics, Zhejiang University, Hangzhou 310027, People's Republic of China
E-mail: songwenhao@msn.com, yxn@zju.edu.cn

Published online: 24 March 2005

A new method of free-form deformation, w-TFFD, is proposed, for which an original shape is deformed using weighted T-spline volumes. We generalize T-splines to weighted T-spline volumes that also permit T-junctions. Weighted T-spline volumes are a natural generalization of NURBS volumes but permit more flexible control lattices. Thus, w-TFFD holds many virtues of traditional FFDs and is more adaptive to objects with arbitrary topology or complex shape. The lattices can be automatically generated and approximate the shape of the object arbitrarily close by octree subdivision.
Besides constructing and deforming a multiresolution lattice, users can also sculpt specific local details to their required shape by modifying weights. A set of direct-acting tools that are similar to previously proposed techniques can be applied to w-TFFD.

Key words: Free-form deformation – T-spline – w-TFFD

1 Introduction

Free-form deformation (FFD), as introduced by Sederberg and Parry [23], is known to be a powerful shape modification method that has been extensively applied to computer animation [4, 7, 14] and geometric modeling [8, 11]. This technique deforms an object by embedding it within a solid defined with a control lattice. A change of the lattice deforms the solid and hence the object. People have proposed various FFD methods. All of these methods solve the main problems of local and global free-form deformation of an object. The matters for FFD are the form of control points, parametrization and user manipulation. However, few of the existing methods satisfy user demand in all aspects.
In this paper, we present a new free-form deformation method using weighted T-spline volumes to provide users with fine control lattices and flexible ways of deformation. We name this method w-TFFD. The main contributions of this paper are as follows:

1. Definition of weighted T-spline volumes and using weighted T-splines as parametric volumes for free-form deformation;
2. Automatic generation of multiresolution lattices for weighted T-spline volumes;
3. Direct deformation for arbitrary shape by combining weights change and lattice manipulation.

Compared with previous FFD methods, the main advantage of the proposed method is that any complex shapes can be approximated by multiresolution control lattices tightly and automatically. Combining weights with the lattices, w-TFFD provides an easy-to-use interface.
T-splines were proposed by Sederberg et al. [24]. T-spline control grids permit T-junctions, so lines of control points need not traverse the entire control grid [24]. Weighted T-spline volumes inherit this virtue. Therefore, control lattices for w-TFFD are more flexible than for traditional FFDs [10, 16, 23]. Moreover, the generation of control lattices is not complex and we adopt octree subdivision for initial lattice construction. Weighted T-spline volumes are C^2 continuous. With the technique of octree subdivision and T-junctions, the control lattice approximates the shape of an object efficiently while the internal lattice has been simplified effectively. We need only manipulate the control points that define nonempty cells (see Fig. 1b) in the octree subdivision lattice for deformation. The rest of the control points are hidden for users, since their influence on shape change is always negligible when their weights are set to

The Visual Computer (2005)
Digital Object Identifier (DOI) 10.1007/s00371-004-0277-8

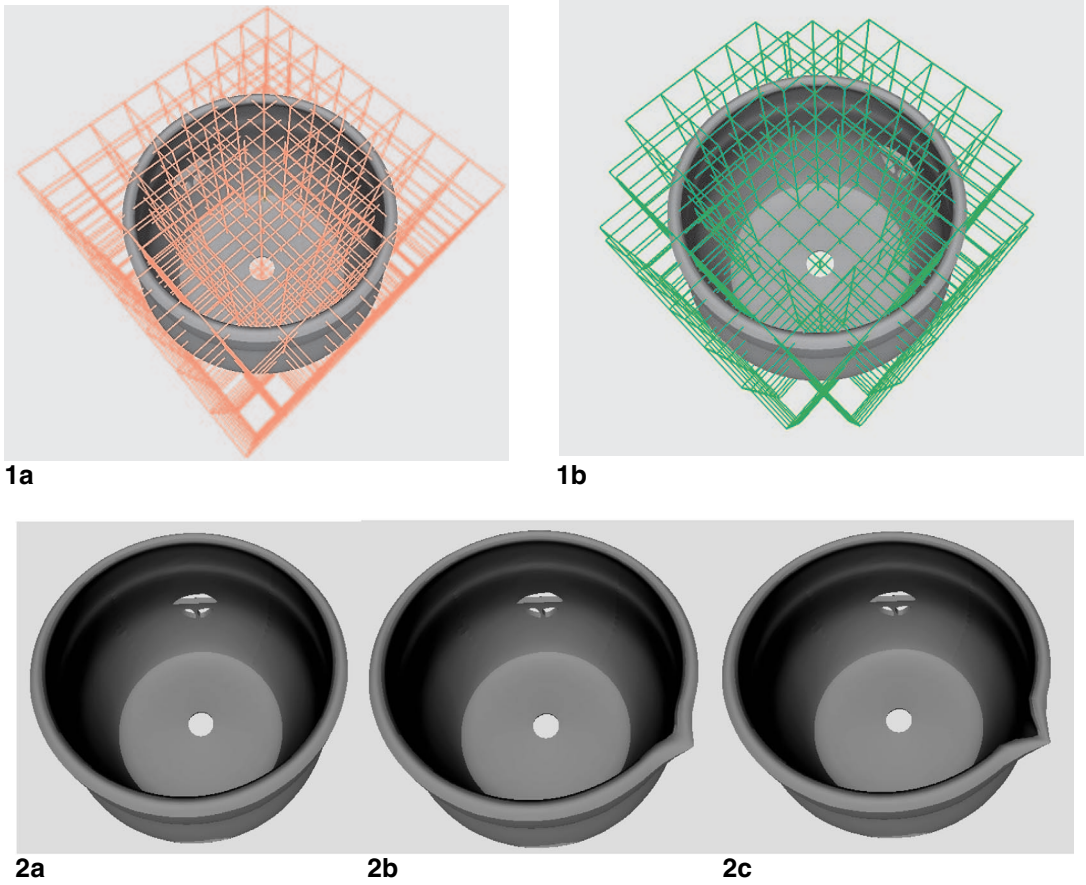


Fig. 1a,b. Octree subdivision lattice generated for a pot. **a** The whole control lattice; **b** The control points used for deformation interference

Fig. 2a–c. The hammer tool acting on a pot. **a** Original mesh; **b**, **c** Sculpted mesh

very small values. The whole lattice (see Fig. 1a) is used for parametrization and relocation. Users can do any level of deformation with ease and intuition. Figure 1 shows two examples with the whole control lattice and the control points belonging to nonempty cells. We can see that the internal control points of the pot have been reduced greatly and the control points used for deformation interference are even less (see Fig. 1).

w-TFFD can also be used to sculpt objects directly by changing weights. Users can interact with w-TFFD through a set of direct-acting tools such as the hammer tool and crimp tool [21], removing the need to manipulate the mesh directly. The hammer tool changes the height of a peak without changing its shape and the crimp tool changes the shape of a peak without altering its height [21]. Those tools that only need two weights across the object for deformation can be applied to w-TFFD. Analysis of the technique is similar to Noble et al. [21]. It provides users a new method of free-form deformation. Figure 2 shows the effect of the hammer tool on a pot.

The rest of the paper is organized in the following manner. In Sect. 2 we introduce some related work. Section 3 briefly reviews T-splines [24] and presents weighted T-spline volumes. Section 4 shows how to generate the T-lattice. Section 5 solves the inverse point problem, that is, parametric coordinates for each point. Section 6 shows the deformation algorithm. Section 7 describes the implementation and

discusses the performance. Conclusions and future work are given in Sect. 8.

2 Related work

The pioneering FFD method proposed by Sederberg and Parry [23] restricts the parametric volume to a regular parallelepiped with uniform divisions and Bernstein polynomial bases. Davis and Burton used rational Bernstein bases [8] and later Griessmair and Purgathofer used a trivariate B-spline [10]. Lamousin and Waggenspack used NURBS volumes [16]. Hsu, Hughes, and Kaufman [12] proposed an FFD that allowed users to manipulate points on the surface of the embedded object directly. However, all of these methods require that the control lattice for deformation should be a regular parallelepiped or a uniformly arranged shape. This requirement makes it difficult to approximate objects with arbitrary shapes.

People also proposed many FFDs that use control lattices with arbitrary topology. In some of these FFDs, the control lattice can be automatically generated. Coquillart [6] presented a more robust version of FFD that did not restrict the enclosing solid to a parallelepiped solid. Arbitrarily shaped lattices are built by combining several tricubic Bézier volumes. It needs to keep the continuous constraints between Bézier volumes and solve the inverse-point problem [6]. Thus, it incurs the significant computational expense of resolving the inverse-point problem. MacCracken and Joy [18] extended FFD further by using subdivision technique, which is a volume extension version of Catmull–Clark subdivision surfaces. Their control lattice has arbitrary topology, which can approximate the object shape more tightly. However, in each subdivision step, a cell of the lattice is subdivided into eight subcells. Obviously, it costs too much space and time.

Bechmann et al. [1] proposed continuous FFD (CFFD). CFFD is based on barycentric coordinates and Bézier tetrahedrons. Combining tetrahedrons allows one to build control lattices of any shape, but keeping the deformation continuous between tetrahedrons requires defining constraints on the displacement of control points.

Dirichlet FFD [20] is an approach based on the Voronoi structure defined within the convex hull of a set of points. While there is no restriction on the shape of the volume, the deformations are controlled solely by the parametrization defined by natural neighbor interpolants. These interpolating functions have singularities that result in unwanted deformation artifacts. In DFFD, the influence is predefined by the control lattice set. It is hard to manipulate the influence of a single control point.

Ono et al. [22] proposed an FFD method with automatic generation of multiresolution lattices. They employ the Catmull–Clark subdivision method for lattice generation, and then parameterize the minimal cell. The method must save topological information of the Catmull–Clark subdivision volumes, so it is memory intensive and can subdivide only a few steps for most practical objects.

FFDs based on two-dimensional manifold control meshes have recently been proposed. Shao et al. [25] presents a FFD method by using arbitrary topological meshes. Kazuya et al. proposed t-FFD [15], which uses a set of triangles with arbitrary topology and geometry as the control mesh. The control mesh can be generated from the original shape or be defined manually. Both methods show that the shape of the control mesh is approximating the shape of the model. However, they can hardly generate multiresolution control mesh on a single hierarchical "level."

Recently skeleton-driven free-form shape deformations have drawn much attention [2, 3, 17, 26, 27] because they are well suited for large-scale shape deformations and, therefore, can be used in numerous applications in the computer game industry. However, these methods are not suitable for local deformation.

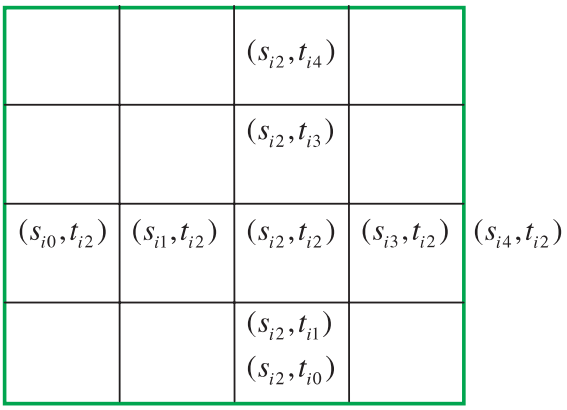
Based on this background, we propose w-TFFD by extending the previous FFDs and providing a more flexible and adaptive FFD method. The details are given in the following sections.

3 T-splines and weighted T-spline volumes

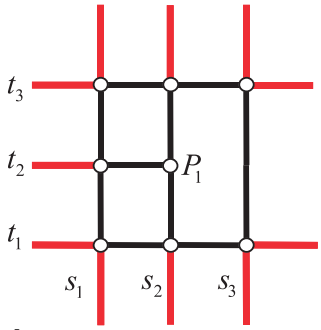
3.1 T-splines

T-splines were introduced by Sederberg et al. in [24]. A T-spline is a PB-spline for which some order has been imposed on the control points by means of a control grid called a T-mesh. First, a PB-spline is defined as

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i B_i(s,t)}{\sum_{i=1}^{n} B_i(s,t)} \quad (s,t) \in D\,, \tag{1}$$



3



4

Fig. 3. Knot lines for basis function $B_i(s, t)$
Fig. 4. T-junction P_1

where the P_i are control points. The $B_i(s, t)$ are basis functions given by

$$B_i(s, t) = N^3_{i0}(s)N^3_{i0}(t), \tag{2}$$

where $N^3_i(s)$ is the cubic B-spline basis function associated with the knot vector

$$s_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}] \tag{3}$$

and $N^3_i(t)$ is associated with the knot vector

$$t_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}] \tag{4}$$

as illustrated in Fig. 3.

A PB-spline is point-based instead of grid-based and satisfies the convex-hull property. Every control point has its influence domain $D_i = (s_{i0}, s_{i4}) \times (t_{i0}, t_{i4})$. The knot vectors s_i and t_i for each basis function are deduced from the T-mesh. A T-mesh is simply a rectangular grid with T-junctions and is constrained by two rules. Each edge in a T-mesh (see Fig. 4) is a line segment of constant s (which is called an s-edge) or of constant t (which is called a t-edge) [24]. A T-junction is a vertex shared by one s-edge and two t-edges, or by one t-edge and two s-edges. For example, P_1 (see Fig. 4) is a T-junction. The details can be found in [24].

T-splines permit T-junctions [24]. It is very useful to build multiresolution control grids, which are denser on the boundary of the object (see Fig. 5). The control grid still satisfies the two rules and is a T-mesh [24]. When T-splines are generalized to 3D space, it is also possible to build multiresolution lattices.

3.2 Weighted T-spline volumes

T-splines [24] can be easily generalized to weighted T-spline volumes. A weighted T-spline volume is defined as a weighted 3D PB-spline for which some order has been imposed on the control points by means of a T-lattice. We first define weighted 3D PB-splines.

3.2.1 Weighted 3D PB-splines

We define the equation for a weighted 3D PB-spline as

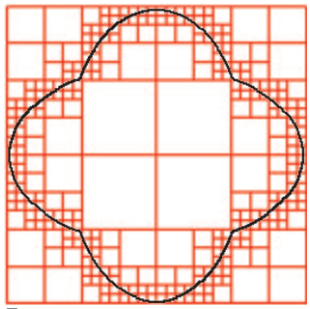
$$P(u, v, w) = \frac{\sum_{i=1}^{n} \omega_i P_i B_i(u, v, w)}{\sum_{i=1}^{n} \omega_i B_i(u, v, w)} \quad (u, v, w) \in D\,, \tag{5}$$

where the P_i are the control points and the ω_i are the positive weights. The $B_i(u, v, w)$ are basis functions given by

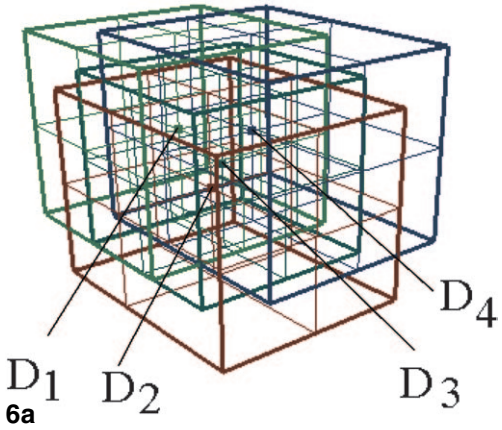
$$B_i(u, v, w) = N^3_{i0}(u)N^3_{i0}(v)N^3_{i0}(w), \tag{6}$$

where $N^3_i(u)$, $N^3_i(v)$ and $N^3_i(w)$ are the cubic B-spline basis functions associated with the knot vectors

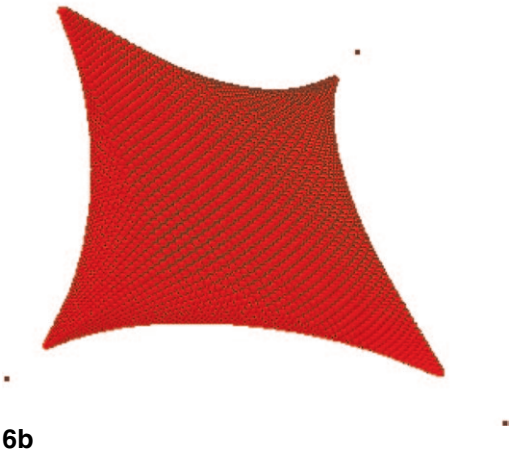
$$\xi_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}], \tag{7}$$



5



6a



6b

Fig. 5. The contour and the control grids
Fig. 6a,b. A 3D PB-spline with four control points. **a** A 3D PB-spline domain; **b** The 3D PB-spline volume

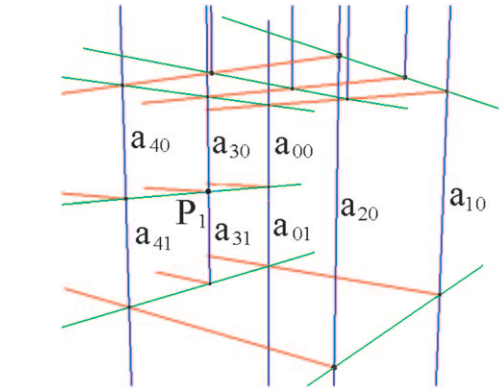
for $\xi = u, v, w$, respectively. The influence domain of $B_i(u, v, w)$ is $D_i = (u_{i0}, u_{i4}) \times (v_{i0}, v_{i4}) \times (w_{i0}, w_{i4})$ and it is clear that D_i is a regular subdivided lattice.

$B_i(u, v, w)$ and its first and second order derivatives all vanish on the bounding box of (and outside of) the D_i. It can be easily proved that weighted 3D PB-splines satisfy the convex-hull property. D in Eq. (5) is the domain over which the entire weighted 3D PB-splines are defined. Weights are introduced for the purpose of providing a new deformation way by changing weights. If all weights are equal, weighted 3D PB-splines reduce to 3D PB-splines. The restriction on D is that $\Sigma_{i=1}^{n} B_i(u, v, w) > 0$ for all $(u, v, w) \in D$ and D is a single connected component. It is advisable to have each point in D lie in at least four influence domains D_i.

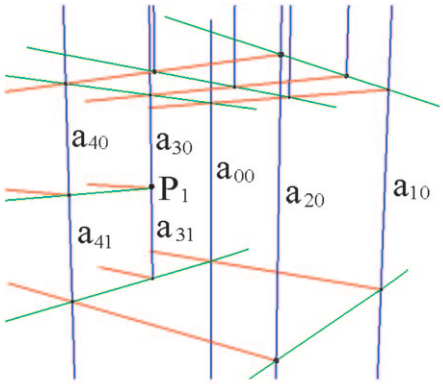
Figure 6a shows a parameter space in which D_i has been drawn for a 3D PB-spline comprised of four blending functions. The resulting volume is shown on the right. A possible choice for D is the interval contained in the intersection of D_is. Like T-splines [24], a weighted 3D PB-spline has no notion of a control lattice. The knot vectors for each basis function are completely independent of the knot vectors for any other basis function.

3.2.2 T-lattice

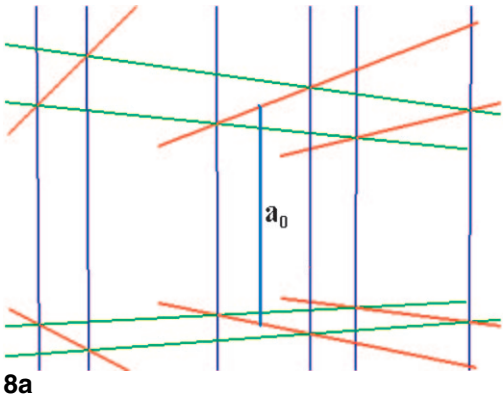
A T-lattice is a rectangular parallelepiped, that allows T-junctions. Like a T-mesh [24], a T-lattice serves two purposes. Firstly, it provides a friendlier user interface than the control points does by an arbitrary weighted 3D PB-splines. Secondly, the knot vectors u_i, v_i, w_i for each basis function are deduced



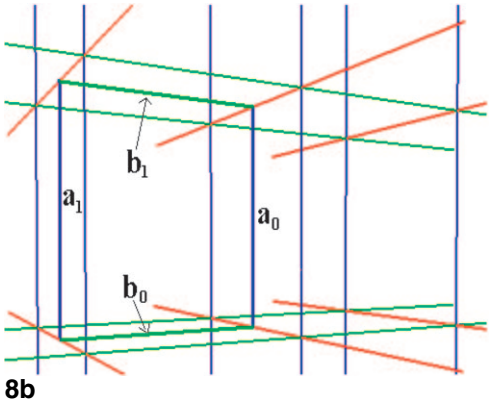
7a



7b



8a



8b

Fig. 7a,b. A minimal cell of a T-lattice. **a** T-junction P_1 shared by one edge in one direction and two edges in the other two directions; **b** T-junction P_1 shared by one edge in two directions and two edges in the other direction

Fig. 8a,b. An edge in a T-lattice. **a** Edge a_0 is not a cell edge; **b** Edge a_0 is a cell edge

from the T-lattice. If a T-lattice is simply a rectangular parallelepiped with no T-junctions, the weighted T-spline volume reduces to a NURBS volume. Further, if all weights are equal, the weighted T-spline volume reduces to a B-spline volume. For example, Fig. 7 shows a minimal cell of T-lattice in (u, v, w) space, and each color denotes an axis.

We call line segments of three directions u-edge, v-edge, and w-edge, respectively. A T-junction is a vertex shared by one edge in some direction and two edges in other directions at the same time. For example, P_1 in Figs. 7a and b are T-junctions. Each edge in a minimal cell of T-lattice denotes a knot interval constrained by the following rules:

Rule 1. In each minimal cell, the sum of knot intervals in the same direction must be equal.

The rule is analogous to that for T-splines [24]. Thus for the cell in Figs. 7a or b in the vertical direction we have

$$a_{00}+a_{01}=a_{10}=a_{20}=a_{30}+a_{31}=a_{40}+a_{41}, \quad \text{or}$$
$$a_{00}=a_{10}=a_{20}=a_{30}+a_{31}=a_{40}+a_{41}.$$

The constraints for the edges in remaining directions may be deduced similarly.

Rule 2. Any edge must be a cell edge.

In Fig. 8a a_0 is an invalid edge because a_0 is not a cell edge. In Fig. 8b, a_0 is a cell edge, and therefore a_0 is a valid edge.

Rule 3. In our method, there are no zero edges in T-lattices.

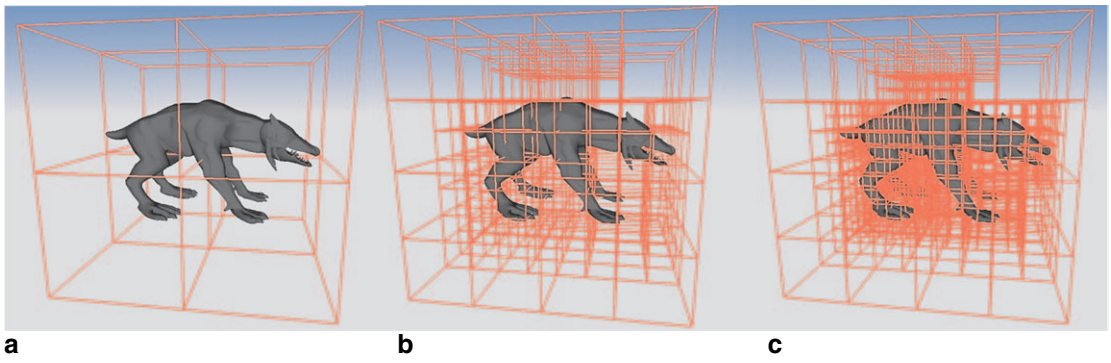


Fig. 9a–c. T-lattice generated by octree subdivision. **a** The initial T-lattice; **b** An intermediate T-lattice; **c** The final T-lattice

From Eqs. (6) and (7) we know that each P_i corresponds to a basis function $B_i(u, v, w)$ defined in terms of knot vectors u_i, v_i, and w_i. We now explain how to infer these knot vectors from the T-lattice. The knot coordinates of P_i are (u_{i2}, v_{i2}, w_{i2}). The knots u_{i3} and u_{i4} are found by considering a ray in the parameter space $R(a) = (u_{i2} + a, v_{i2}, w_{i2})$. Then u_{i3} and u_{i4} are the u coordinates of the first two u-edges intersected by the ray (not including the initial (u_{i2}, v_{i2}, w_{i2})) with $v \times w$ planes. The other knots in v and w directions are found in a similar manner.
Weighted T-spline volumes keep C^2 continuous in the whole parametric volume. Therefore, no continuity problem should be considered for deformation. However, the continuity problem must be handled seriously by some traditional FFDs [1, 6]. It is easy to prove that any octree-subdivided lattice satisfies the above three rules and is a valid T-lattice. Therefore, we can use octree subdivision lattices as T-lattices directly.

4 Automatic generation of T-lattice

Before the generation of a T-lattice, we should first determine the initial region of the model to be deformed. The initial region can be an axis-aligned bounding box (AAB), an oriented bounding box (OBB) [9], or a minimum-volume bounding box. The region may cover the whole model or just some parts of it. Given the region to be deformed, we will make a T-lattice approximating the shape of the object by octree subdivision. The steps for T-lattice generation using octree subdivision are as follows:

1. Define the initial lattice to be deformed.
2. If the cell contains any vertex of the model, subdivide it by applying the octree subdivision.
3. Repeat steps 2 and 3 until a user-specified threshold is reached.

Figure 9 shows the T-lattices, which approximate the shape of the model hierarchically by octree subdivision. In [22], Ono et al. employed Catmull–Clark subdivision to generate the control points and parameterize the minimal cells. The memory required to keep topological information of the Catmull–Clark subdivision volume for parametrization increases by a factor of eight with each subdivision step. Therefore, it is almost impossible to generate high-level subdivision lattices. In this paper, we use weighted T-spline volumes as parametric volumes. It need not keep the topological information of each subdivision level. We only keep the T-lattice subdivided by octree. The data structure of the octree is simple and consumes less memory. When the control lattice has been subdivided by octree subdivision, it needs much less space than uniformly subdivided lattices. Another virtue is that it greatly reduces the internal control points that cannot be easily seen or used. We can see that only the region close to the object is subdivided at higher level (see Fig. 9), and the control points are mostly concentrated on the boundary of the object.
The initial weights of control points belonging to nonempty cells are set much larger than the other

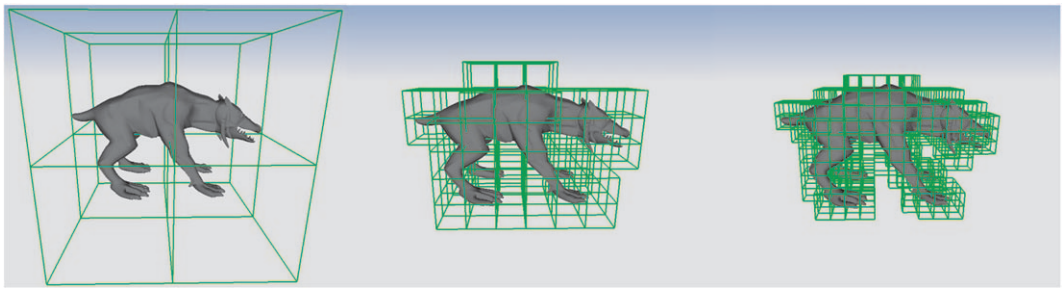


Fig. 10a–c. Control points belonging to the nonempty cells. **a** The initial lattice; **b** An intermediate lattice; **c** The final lattice for deformation

ones. The ratio of them is 100:1. Therefore, the control points close to the object weigh heavily and have evident influence on the small region containing the boundary. We can achieve the deformation by manipulating the control points only close to the shape. Thus, we use only the control points that belong to the nonempty cells for deformation and hide the rest of the control points from user interference (see Fig. 10). We will show that w-TFFD is well-suited for local and global deformation.

We make it clear that the control lattice can be subdivided into arbitrary topological structure as long as it satisfies the three rules presented in Sect. 3.2.2. In our paper, we use octree subdivision only for the purpose of simpleness and practice.

5 Parametrization

w-TFFD deforms objects by embedding them within the solid and identifies the parametric coordinates associated with the point set that represents the object to be deformed. Therefore, we should calculate the parametric coordinates for each point before deformation. Let $Q = (x, y, z)$ be an arbitrary point of the original object, w-TFFD parameterizes the point Q by the T-lattice which includes Q. With the parameters calculated, the point Q will be mapped to a new point Q' according to the modified lattice or the modified weights. When the object is embedded into the T-lattice, w-TFFD parameterizes the point Q by the following equation

$$Q = \frac{\sum_{i=1}^{n} \omega_i P_i B_i(u, v, w)}{\sum_{i=1}^{n} \omega_i B_i(u, v, w)} \quad (u, v, w) \in D\,. \tag{8}$$

The cells of octree-subdivided lattice are generally not uniform, or aligned with the data axes. Therefore, the equation cannot be solved directly as previously proposed methods [8, 10, 11], nor can the problem be separated into three independent parts, one for each parametric variable like NFFD [16]. Equation (8) is a nonlinear equation and we can solve the equation numerically.

It can be easily verified that the solution to Eq. (8) is unique, and the solution can be obtained by solving an equivalent scalar function $F(u, v, w) = \|P(u, v, w) - Q\|^2 = 0$. Then, we solve the scalar equation by employing nonlinear conjugate gradient method [13]. Let $q_k = (u_k, v_k, w_k)$ be the approximate solution after k times of calculation and d_k be the descent direction at q_k, then the solution can be improved further by searching along the direction d_k. With this search, we have $q_{k+1} = q_k + \alpha_k d_k$, and α_k satisfies the strong Wolfe conditions [13]:

$$F(q_k + \alpha_k d_k) \le F(q_k) + c_1 \alpha_k \nabla F^T(q_k) d_k \tag{9}$$

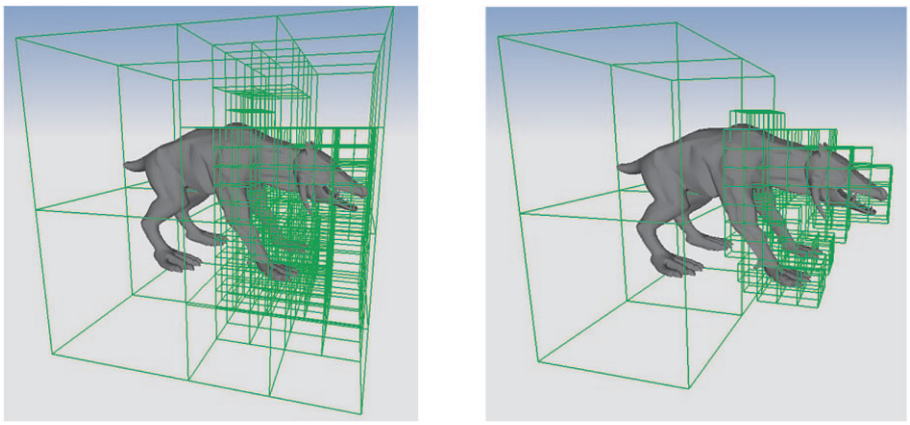
$$|\nabla F(q_k + \alpha_k d_k)^T d_k| \le c_2 |\nabla F(q_k)^T d_k| \tag{10}$$

where $0 < c_1 < c_2 < 1/2$ and $\nabla F_k = (F_u(u_k, v_k, w_k), F_v(u_k, v_k, w_k), F_w(u_k, v_k, w_k))^T$ is the gradient vector of $F(u, v, w)$ at q_k. With q_{k+1} obtained and for further computation, the descent direction d_{k+1} at q_{k+1} can be computed as

$$d_{k+1} = -\nabla F_{k+1} + \beta_k d_k, \tag{11}$$

where $\beta_k = \frac{\nabla F_{k+1}^T \nabla F_{k+1}}{\nabla F_k^T \nabla F_k}$.

From Eq. (9)–(11), we can see that the gradient of $F(u, v, w)$ should be computed for each iteration. For the purpose of efficiency, we compute



a **b**

Fig. 11a,b. Part of the lattice is subdivided by octree subdivision. **a** The T-lattice by local octree subdivision; **b** The control points belonging to nonempty cells

$F'_u(u, v, w)$, $F'_v(u, v, w)$ and $F'_w(u, v, w)$ before deformation. As to the initial solution and the descent direction, because the parametric domain D is identical with the bounding box of the T-lattice, we choose the initial solution $q_0(u_0, v_0, w_0)$ equal to the point Q itself and the descent direction $d_0 = -\nabla F_0$.
At present, we cannot prove the convergence of the numerical algorithm theoretically. If the solution does not converge within limited steps, we can reset a new initial solution and solve the equation again. Because weighted T-spline volumes are not very distorted or eccentric, we have obtained all solutions without any recomputation. In fact, when we choose the precision $|P(q_k) - Q| < 0.001$, it needs no more than 15 iterations to solve the equation.

6 Deformation algorithm

With the construction of weighted T-spline volumes and by manipulating the control lattice or the weights of T-spline, we can then deform any object embedded in the volume. The main steps for the w-TFFD algorithm are as follows:

Step 1. Define the initial region of the model to be deformed.

Step 2. Generate the multiresolution lattices and set initial weights for T-spline volumes.

Step 3. Calculate the parametric coordinates (u, v, w) for each point.

Step 4. Modify the control points or weights and evaluate the new locations of the points.

In step two, users can only apply octree subdivision to user-specified cells. Figure 11 shows that part of the lattice has been subdivided. It is similar to the local subdivision proposed in [19]. The result of the control lattice still satisfies the three rules proposed in Sect 3.2.2 and is a valid T-lattice (see Fig. 11a).
w-TFFD supports direct manipulation. Some direct acting tools, such as the hammer tool and crimp tool proposed in [21], can also be applied to w-TFFD, even though there are still some differences between their methods of manipulation. Because the control lattice of w-TFFD needn't be aligned with the data axes, we cannot operate one row of the control lattice in some directions as large tools [21] do. In our method, we only need to operate the weights of those control points belonging to nonempty cells. From Eq. (5), it is clear that increasing a weight at a point, will move all affected points within the volume towards that point. It is similar to the result proposed by Noble et al. [21].

7 Discussions and implementation results

In this section, we present several examples to show the efficiency of global and local deformation by w-TFFD.

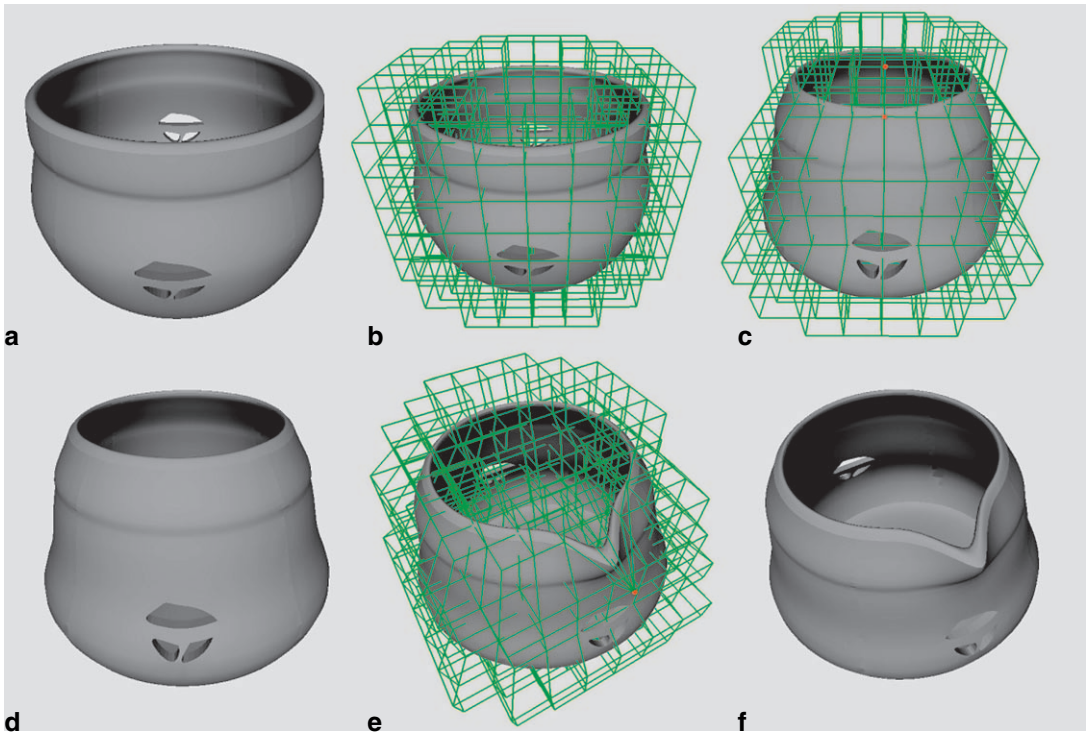


Fig. 12a–f. "Pot" example: global and local deformation. **a** The original mesh of pot; **b** The control lattice; **c** Deformation by changing control lattice; **d** The result of **c**; **e** Adding local details; **f** The result of **e**

The first example is for "Pot" deformation (see Fig. 12). It demonstrates global and local deformation by w-TFFD. The entire shape of the pot is modified by manipulating all the control points. The user can easily do such global deformation with less control points, pulling the rim of the pot out by varying the weights and displacements of the control points marked red. In Fig. 12e, we combine the two ways of deformation and achieve a better effect with ease.

The second example, "Tree" deformation (see Fig. 13), demonstrates deformation of objects with complex geometry. Octree subdivision can be used to approximate the shape of the tree efficiently and all tree branches are deformed in a single w-TFFD. However, some traditional FFDs, using Bézier volumes, B-spline volumes, or NURBS volumes, will have to separate the deformation into several independent ones. These methods then suffer either high computational cost or great inconvenience.

The last example, "Rhino" deformation (see Fig. 14) demonstrates deformation with local subdivision. The control lattice of each part is subdivided into different levels and that of the Rhino head is subdivided at a higher level. Therefore, we can do finer deformation on the head.

We evaluate the performance of w-TFFD system by calculating the number of control points and the parametrization time, on a PC with Pentium-IV 2.8 GHz CPU and 1 GB memory.

The results are shown in Table 1, including a comparison between the numbers of control points generated by octree subdivision and by regular subdivision. From Table 1, we see that with the same depth of subdivision, the new method can be used to reduce the control points greatly. Here the control points are the whole points for T-lattice. If we preclude the control points that are not manipulated by users, the number will be even less. So our method provides easy manipulation for users. The memory that w-TFFD requires is only used to store the control lattice and the octree. It consumes less memory than regular subdivision. The

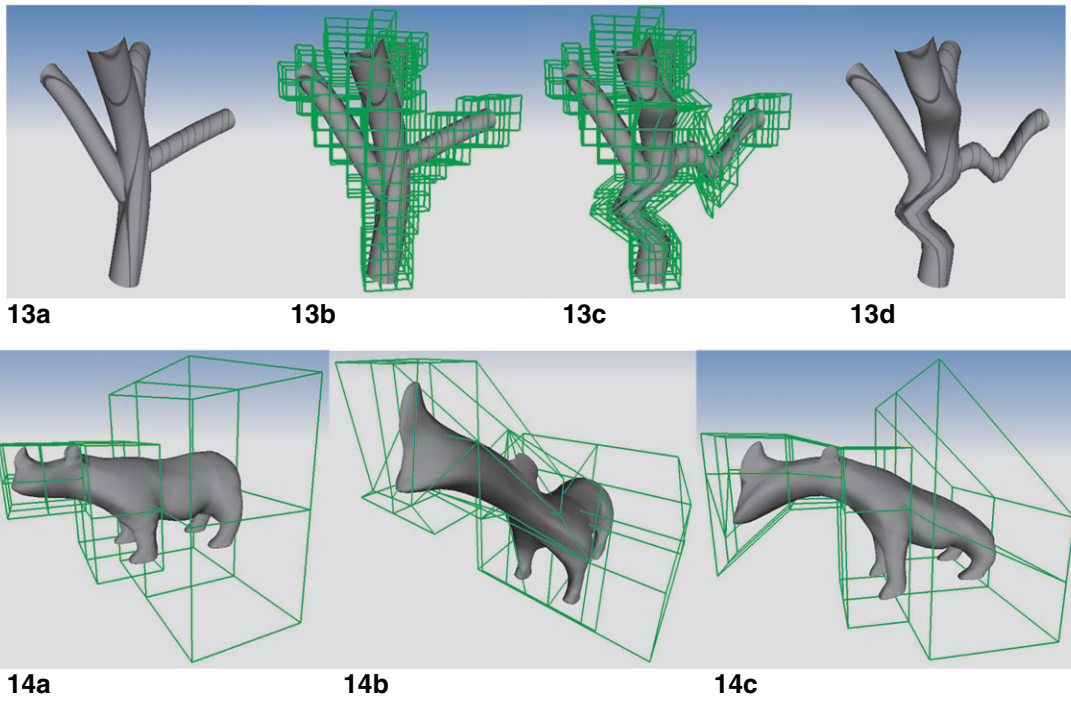


Fig. 13a–d. "Tree" example: deformation of objects with complex topology. **a** The original mesh of tree; **b** The control lattice; **c** Deformation with lattice; **d** The result of **c**

Fig. 14a–c. "Rhino" example: local subdivision. **a** The original mesh of rhino; **b**, **c** Deformation results

most time-consuming step is the process for calculating the parametric coordinates. Obviously, the parametrization time depends heavily on the point number of models, as well as the octree depth. In this paper, we solve the nonlinear Eq. (8) by employing nonlinear conjugate gradient method. The efficiency depends greatly on the initial solutions. We can use heuristic evaluation by the point's adjacency to reduce the time. The times for multiresolution lattice generation and point relocation are negligible.

8 Conclusions and future work

From all the examples we have presented, we can see that w-TFFD has greater deformation ability than previous FFDs. Moreover, it combines more defor-

Table 1. Performance of w-TFFD. CPNOS: the number of control points generated by octree subdivision; CPNRS: the number of control points generated by regular subdivision; PT: the parametrization time

Model (number of points)	Octree depth	CPNOS	CPNRS	PT (s)
Tree (8738)	3	346	729	62.5
	4	1141	4913	97.3
	5	4002	35937	162.5
Tusker (9555)	3	421	729	83.2
	4	1556	4913	122.5
	5	5725	35937	240.3
Pot (14737)	3	678	729	131.2
	4	3511	4913	210.7
	5	15825	35937	405.7

mation techniques than previous FFDs in a consistent framework. The main features of w-TFFD are that it provides multiresolution lattices and supports direct manipulation in a consistent framework. The control lattice can be automatically generated and approximate arbitrary shape more tightly with less memory increase. People can do any level of deformation and manipulate the control points close to the shape easily. Due to direct manipulation by changing weights, potential difficulties of the FFD mentioned in [21] can be avoided, and some tools based on NURBS [21] can be applied to w-TFFD and provide greater versatility of sculpting.
Several desirable extensions need to be studied in the future. These include:

- To generate control lattices with arbitrary topology that is better than octree subdivision lattices;
- To support hierarchical deformation;
- To import T-spline surfaces by creating a T-spline volume in which the surface is an isosurface; and
- To implement w-TFFD by hardware acceleration as proposed in [5].

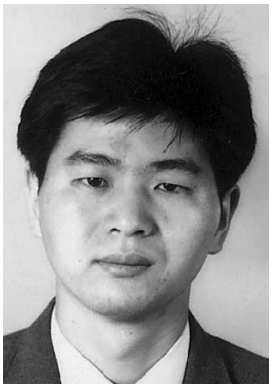
Acknowledgements. This work was supported by NSFC grants 60303015 and 60333010 and the 973 program grant 2002CB312101.

References

1. Bechmann D, Bertrand Y, Thery S (1996) Continuous free form deformation. Comput Netw ISDN Syst 29(14):1715–1725
2. Bloomenthal J (2002) Skinning: medial-based vertex deformation. In: Proceedings of ACM SIGGRAPH Symposium on Computer Animation, pp 147–151
3. Bloomenthal J, Lim C (1999) Skeletal methods of shape manipulation. In: Proceedings of Shape Modeling International '99, pp 44–47
4. Chadwick JE, Haumann DR, Parent RE (1989) Layered construction for deformable animated characters. ACM Comput Graph (SIGGRAPH '89) 23(3):243–252
5. Chua C, Neumann U (2000) Hardware-accelerated free-form deformation. In: Proceedings of ACM SIGGRAPH Symposium on Graphics hardware, pp 33–39
6. Coquillart S (1990) Extended free-form deformation: a sculpturing tool for 3D geometric modeling. ACM Comput Graph (SIGGRAPH '90) 24(4):187–193
7. Coquillart S, Jancene P (1991) Animated free-form deformations: an interactive animation technique. ACM Comput Graph (SIGGRAPH '91) 25(4):23–26
8. Davis OR, Burton RP (1991) Free-form deformation as an interactive modeling tool. Journal of Imaging Technology 17(4):181–187
9. Gottschalk S, Lin MC, Manocha D (1996) OBB-Tree: a hierarchical structure for rapid interference detection. In: Proceedings of ACM SIGGRAPH 96 Conference, pp 171–180
10. Griessmair J, Purgathofer W (1989) Deformation of solids with trivariate B-splines. In: Proceedings of Eurographics'89. Elsevier, North Holland, pp 137–148
11. Gudukbay U, Ozguc (1990) Free-form solid modeling using deformation. Comput Graph 14(3/4):491–500
12. Hsu W, Hughes J, Kaufman H (1992) Direct manipulation on free-form deformation. ACM Comput Graph (SIGGRAPH'92) 26(2):177–184
13. Jorge N, Stephen JW (1999) Numerical optimization. Springer, Berlin Heidelberg New York
14. Karla P, Mangli A, Thalmann NM, Thalmann D (1992) Simulation of facial muscle actions based on rational freeform deformation. Comput Graph Forum (Eurographics'92), pp 59–69
15. Kazuya G, Kobayashi, Katsutoshi O (2003) t-FFD: free-form deformation by using triangular mesh. In: Proceedings of the eighth ACM symposium on solid modeling and applications, pp 226–234
16. Lamousin H, Waggenspack W (1994) NURBS-based free-form deformation. IEEE Comput Graph Appl 14(9):59–65
17. Lewis JP, Cordner M, Fong N (2000) Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In: proceedings of ACM SIGGRAPH, pp 165–172
18. MacCracken R, Joy K (1996) Free-form deformation with lattice of arbitrary topology. In: Proceedings of ACM SIGGRAPH, pp 181–188
19. McDonnell KT, Qin H, Wlodarczyk RA (2001) Virtual clay: a real-time sculpting system with haptic toolkits. In: Proceedings of ACM Symposium on Interactive 3D Graphics, pp 179–190
20. Moccozet L, Thalmann NM (1997) Dirichlet free-form deformations and their application to hand simulation. In: Proceedings of Computer Animation'97, IEEE Computer Society Press, pp 93–102
21. Noble RA, Gordon JC (1999) Direct manipulation of surfaces using NURBS-based free-form deformations. In: Proceedings of the International Conference on Information Visualization
22. Ono Y, Chen B, Nishita T, Feng J (2002) Free-form deformation with automatically generated multiresolution lattices. In: Proceedings of the First International Symposium on Cyber Worlds, pp 472–490
23. Sederberg T, Parry S (1986) Free-form deformation of solid geometric models. ACM Comput Graph (SIGGRAPH'86) 20(4):151–160
24. Sederberg T, Zheng J, Bakenov A, Nasri A (2003) T-splines and T-NURCCs. ACM Trans Graph 22(3):477–484
25. Shao J, Zhao Y, Feng J, Jin X, Peng Q (2003) Free-form deformation by using arbitrary topological mesh. In: Proceedings of CAD&Computer Graphics, 29–31 October 2003, Macao
26. Singh K, Kokkevis E (2000) Skinning characters using surface-oriented free-form deformations. In: Proceedings of Graphics Interface pp 35–42
27. Yoshizawa S, Belyaev AG, Seidel HP (2003) Free-form skeleton-driven mesh deformations. In: Proceedings of the eighth ACM symposium on solid modeling and applications, pp 247–253

Photographs of the authors and their biographies are given on the next page.

WENHAO SONG graduated from Zhejiang University, China in 2002. Currently he continues in the MS program at Zhejiang University. His major is applied mathematics and he is doing research on computer graphics.



XUNNIAN YANG is now an associate professor in the Department Of Mathematics and Institute of Computer Graphics and Image Processing at Zhejiang University, China. He obtained a BS in applied mathematics from Anhui University and a PhD in CAGD and computer graphics from Zhejiang University in 1993 and 1998, respectively. His research interests include computer aided geometric design and computer graphics.