ELSEVIER

Computer Aided Geometric Design 17 (2000) 59–81

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/comaid

Volume morphing and rendering—An integrated approach

Shiaofen Fang [a,*], Rajagopalan Srinivasan [b], Raghu Raghavan [c], Joan T. Richtsmeier [d]

[a] *Department of Computer and Information Science, Indiana University, Purdue University, Indianapolis, IN 46202, USA*

[b] *Kent Ridge Digital Labs, Singapore, Singapore*

[c] *Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA*

[d] *School of Medicine, The Johns Hopkins University, Baltimore, MD 21205, USA*

Received May 1997; revised April 1999

Abstract

In this paper, we first introduce a 3D morphing method for landmark-based volume deformation, using various scattered data interpolation schemes. Qualitative and speed comparisons are also made for different interpolation schemes. To efficiently render the volume morphing process, a new deformable volume rendering algorithm is presented. The algorithm renders the deformed volume directly without going through the expensive volume construction process. Piecewise linear approximation of the deformation function by adaptive space subdivision and template-based block projection are used to speed up the rendering process. While the resultant timings is slower than real time, it is much faster than existing volume morphing/rendering pipelines. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: 3D Morphing; Volume rendering; Visualization; Scattered data interpolation; Raycasting

1. Introduction

1.1. Volume morphing and volume rendering

A volume dataset is a regular 3D array of sampling data in a scalar or vector field. It is the most common data format in medical imaging data acquisition techniques, such as computed tomography (CT) and magnetic resonance imaging (MRI).

* Corresponding author. E-mail: sfang@cs.iupiu.edu.

PII: S0167-8396(99)00039-4

Visualization techniques for such volume datasets, have been extensively studied and widely used in medical imaging and scientific visualization. In this paper, we will study two interrelated problems that have not been receiving sufficient research attention: landmark-based volume morphing, and the volume rendering of the morphing process.

Volume morphing is a technique for generating smooth 3D image transformation and deformation. Its applications include the creation of new forms, morphological study of biological systems, and analysis and simulation of shape deformation due to growth, evolution, or surgical reconstruction. In craniofacial surgery planning, for instance, morphing techniques can be used to simulate the post-operative growth of the pediatric patients to provide information about their future appearances. Morphing between the heads of different people and species provides information and tools for the study of the morphological and physiogenetic relationships between people and across species. It also has potential applications in entertainment industry.

Volume rendering, on the other hand, is widely used in visualizing the internal structures of large and complex volumetric objects. Rendering the volume deformation process using regular volume rendering algorithms is, however, very difficult since it requires the construction of all the intermediate deformed volumes. Such volume construction is often very expensive, and far from being real-time or interactive. In this paper we will develop a new deformable volume rendering algorithm that avoids the deformed volume construction process and significantly improves the rendering speed.

1.2. Related work

Earlier work in volume morphing includes the scheduled Fourier method (Hughes, 1992) and the wavelet-based method (He et al., 1994). Only the morphing of simple objects were shown in the above two methods, and non-smooth movements of iso-surfaces have also been observed in the results from these two methods. A more recent work on feature based volume morphing by Lerios et al. (1995), is a 3D extension of a 2D feature-based image morphing algorithm (Beier and Neely, 1992). This method shows much more complicated and smoother results for both the CT volume datasets and polygonal objects. The quality of its morphing results, however, largely depend on the placement of the feature landmarks. More discussions on this work will be given later in Section 4.

2D image morphing, often called image warping, constructs image sequences showing a progressive transition from one image to another. The study of such 2D morphing techniques has been very extensive (Arad et al., 1994; Beier and Neely, 1992; Lee et al., 1995; Ruprecht and Muller, 1995; Wolberg, 1990). Besides its visual drawbacks, such as the difficulties in handling viewing and lighting parameters and spatial relationships, 2D image warping cannot generate the morphed 3D models that are needed for non-visual applications such as analysis and measurements. For 3D volumetric data, a true 3D morphing technique is clearly more desirable.

There are a large number of publications on the topic of volume rendering. They can be roughly classified into two categories: image space approaches such as raycasting based algorithms (Danskin and Hanrahan, 1992; Levoy, 1990; Yagel and Kaufman, 1992), and

object space approaches (Laur an Hanrahan, 1991; Tost et al., 1995; Wilhelms and Gelder, 1991). They are, however, all based on regular volume datasets. Therefore, in order to render a deformed volume, a new regular volume dataset has to be constructed, which can take hours for a typical 256^3 volume (Fang et al., 1996a; Lerios et al., 1995). Even the approximated method proposed in (Lerios et al., 1995) still needs several minutes to form one such volume. To avoid resampling in volume reconstruction, another solution is to generate an irregular volume dataset by direct deformation of the voxels of the source volume, and then use irregular data volume rendering techniques (Koyamada and Ito, 1995; Silva et al., 1996; Wilhelms et al., 1996) to render the deformed irregular volume. This, of course, still requires a volume reconstruction process, and irregular volume rendering algorithms are also much slower. Another relevant work is the ray deflector based raycasting by Kurzion and Yagel (1997), in which the deformation is modeled by ray deflectors and rendered with deflected curved rays to generate visual deformation effects. More discussions on this work will also be given later in Section 4.

1.3. Overview

We start, in Section 2, with a description of various volume morphing methods. We are mainly interested in landmark-based morphing, in which the morphing transformation is guided by the interpolation of a set of landmark points. In biomedical applications, landmarks are homologous biological points that can be defined with high precision on objects of interest. Since the landmarks are often irregularly distributed across the volume, a scattered data interpolation problem has to be solved. Two classes of interpolation methods, the Shepard-based methods and the radial basis function methods, are explored and studied for volume morphing application.

In Section 3, the deformable volume rendering algorithm is presented. It is an object space algorithm based on octree block projection. For each given pair of landmark sets in the original and deformed (target) volumes, the algorithm first generates a backward morphing function from the target volume to the original volume. An adaptive subdivision process using octree encoding will then select target octree blocks within which the deformation function can be approximated by trilinear interpolation. Thereafter, for each viewing direction, the target blocks are projected to the projection plane in a front-to-back order. Each block projection is essentially a mini-raycasting process involving ray-block intersections, and color/opacity compositing and blending. If we use only parallel projections, the block projection process can be further optimized by taking advantage of the uniform sizes and orientation of the target blocks: the ray-block intersection only needs to be done once for sample block of a given size and the result can be “pasted” to all other blocks of the same size. To render the intersecting ray segments, the color/opacity values of the sampling points on the ray segments are obtained from their morphed points in the source volume. Such morphing function evaluation can, however, be approximated by trilinear interpolations of the morphing of their block vertices to improve the rendering speed. The implementation issue of the algorithm will also be discussed, with experimental results and performance data.

More discussions and conclusive remarks are given in Section 4.

2. Landmark-based volume morphing methods

A landmark based volume morphing can be defined as a globally smooth interpolation function between landmark points defined in 3D volume space. Given the source landmark set

$$L^0 = \{L_i^0\} \quad (i = 1, \ldots, n),$$

defined on the source volume V_0, and the target landmark set $L^1 = \{L_i^1\}$ $(i = 1, \ldots, n)$, defined on the target (deformed) volume V_1, a morphing function $F : R^3 \to R^3$ can be generated such that:

$$F(L_i^1) = L_i^0 \quad (i = 1, \ldots, n). \tag{1}$$

This is a backward morphing function that maps points in the target volume to points in the source volume. For each point (or voxel) Q in the target volume, $P = F(Q)$ is its corresponding point in the source volume. Trilinear interpolation can be used to find the voxel value of P in the source volume, which is then assigned back to the voxel Q in the target volume. This process can be repeated for all voxels in V_1 to reconstruct the entire deformed volume.

Since the landmarks are irregularly distributed over the volume space, they are often called *scattered* data. Interpolating such data is called *scattered data interpolation*. This problem has been extensively studied in computer aided geometric design (CAGD) and geometric modeling (Franke and Nielson, 1980; Nielson, 1993). In the following sections, after defining the landmarks, we will introduce various interpolation methods leading to smooth and efficient volume morphing functions.

2.1. Landmarks

Landmarks represent the important geometric and/or biological features of an object. Because of the ease for interactive definition and modification, point landmarks are the most frequently used, particularly in biomedical applications. For example, in craniofacial modeling, three dimensional coordinates of point landmarks from the face, cranial base and neurocranium can be located interactively on the volume or surface rendered images, using an interactive volume visualization tool, for morphological study and growth analysis (Bookstein, 1991; Richtsmeier and Lele, 1993).

Given the source landmark set, $L^0 = \{L_i^0\}$ $(i = 1, \ldots, n)$, the target landmark set $L^1 = \{L_i^1\}$ $(i = 1, \ldots, n)$, can be defined in one of the following ways:

(1) Interactively move the source landmarks with an interactive volume visualization tool. This can be the same program used for picking the source landmarks in the source volume.
(2) Picking the corresponding landmark set in a different volume. This can be useful if we want to study the morphological relationship between two existing volume datasets or explore the processes responsible (e.g., growth, evolution) for that relationship.
(3) The target landmark set can be computed based on certain deformation scheme. An important example is in growth prediction, where an underlying growth model is

used to move the landmarks according to the predicted growth pattern. One such growth model was given by Richtsmeier and Lele (1993). Based on this model, a growth matrix is first defined to represent the change in the relative location of landmarks between the target and reference form as measured by the change in distances between the landmarks before and after growth. Let $\{r_{ij}\}$ be the growth matrix. We have

$$r_{ij} = \left\| L_i^1 - L_j^1 \right\| / \left\| L_i^0 - L_j^0 \right\|.$$

This growth matrix can be statistically derived from a database of previously recorded growth data. A growth prediction method based on a growth difference matrix analysis is then applied to obtain the coordinates of the grown landmark points. Details of this method can be found in (Richtsmeier and Lele, 1993).

Using the same growth matrix, we also developed an energy minimization approach to compute the target landmarks. This approach assumes that a spring, with the rest length of $r_{ij}\|L_i^0 - L_j^0\|$, is attached to each pair of the original landmarks L_i^0 and L_j^0, causing the landmarks to naturally adjust their coordinates trying to reach an equilibrium. Mathematically, it is equivalent to the numerical minimization of the energy term:

$$E = \sum_{i \neq j} \left| \left(\|L_i - L_j\| / \|L_i^0 - L_j^0\| \right) - r_{ij} \right|^2.$$

If the growth matrix is computed from the same pair of landmark sets, L^0 and L^1, E will converge to zero with $\{L_i\}$ going to $\{L_i^1\}$. Otherwise, $\{L_i\}$ will converge to some values, $\{L_i^1\}$, that minimize E. Apparently, the result of this approach depends on the definition of the energy function E.

2.2. Shepard method

The original Shepard method was first proposed by D. Shepard in (1968). It takes the distance weighted average of the interpolation points as the interpolation function. Directly applying this method to our $R^3 \to R^3$ interpolation problem leads to:

$$F(P) = \sum_{i=1}^{n} \frac{L_i^0}{d_i^r(P)} \Bigg/ \sum_{i=1}^{n} \frac{1}{d_i^r(P)}, \tag{2}$$

where $d_i(P)$ is the distance function from L_i^1 and r is the exponent parameter, often set to the power of 2 to avoid computing the square root. When $r > 1$, $F \in C^1$.

This, however, does not generate a satisfactory morph due to its tendency of moving all points towards the center of the volume (Fang et al., 1996a). A common way to fix this (Franke and Nielson, 1980; Ruprecht and Muller, 1995) is to interpolate the displacements of the landmark points, $\{L_i^0 - L_i^1\}$, instead of the actual points $\{L_i^0\}$. This leads to the first volume morphing method—Shepard method:

$$F_s(P) = P + \sum_{i=1}^{n} \frac{L_i^0 - L_i^1}{d_i^r(P)} \Bigg/ \sum_{i=1}^{n} \frac{1}{d_i^r(P)}. \tag{3}$$

2.3. Locally Affine Shepard method

Morphing generated from $F_s(P)$ has two problems:

(1) The Shepard weight function changes abruptly around landmarks, often resulting in too much distortion in certain areas (e.g., the top of the head and around the mouth area in Fig. 1(c)).

(2) The transformation (3) is not affine invariant, i.e., $\forall$ affine transformation $g : R^3 \to R^3$:

$$F_s\big(P; \{L_i^1\}, \{g(L_i^0)\}\big) \neq g\big(F_s\big(P; \{L_i^1\}, \{L_i^0\}\big)\big).$$

Affine invariance property is very desirable since if the source and the target landmarks are placed independently, they can be in totally different coordinate systems. A degenerate case is that if $\{L_i^0 = g(L_i^1)\}$, F_s is generally not the same as g.

To solve these problems, we generalized the Shepard method by using a more general affine interpolant in Eq. (3). Assuming a local affine interpolant, $g_i : R^3 \to R^3$, exists for each interpolation point L_i^1, i.e., $g_i(L_i^1) = L_i^0$, the new interpolation function is defined as:

$$F_a(P) = \sum_{i=1}^{n} \frac{g_i(P)}{d_i^r(P)} \Big/ \sum_{i=1}^{n} \frac{1}{d_i^r(P)}. \tag{4}$$

An energy minimization process is employed to construct the local interpolants $\{g_i\}$, $(i = 1, \ldots, n)$. Let

$$g_i = A_i(x - x_i) + B_i(y - y_i) + C_i(z - z_i) + L_i^0, \tag{5}$$

where $L_i^1 = (x_i, y_i, z_i)$, and $A_i, B_i, C_i \in R^3$ are 3D coefficients of g_i. (A_i, B_i, C_i) can be computed by the numerical minimization of the energy term:

$$E_i(A_i, B_i, C_i) = \sum_{j \neq i} \frac{(g_i(L_j^1) - L_j^0)^2}{d_i^2(L_j^1)} \Big/ \sum_{j \neq i} \frac{1}{d_i^2(L_j^1)}. \tag{6}$$

It is now easy to verify:

(1) Interpolation condition is satisfied, i.e., $F_a(L_i^1) = L_i^0$.

(2) When $r > 1$, the first derivatives of F_a at all the interpolation points interpolate the first derivatives of the local interpolants, i.e.,

$$\frac{\partial F_a(L_i^1)}{\partial x} = \frac{\partial g_i(L_i^1)}{\partial x} = A_i; \qquad \frac{\partial F_a(L_i^1)}{\partial y} = \frac{\partial g_i(L_i^1)}{\partial y} = B_i;$$
$$\frac{\partial F_a(L_i^1)}{\partial z} = \frac{\partial g_i(L_i^1)}{\partial z} = C_i. \tag{7}$$

(3) F_a is affine invariant, i.e., $\forall$ affine transformation $g(\,)$,

$$F_a\big(P; \{L_i^1\}, \{g(L_i^0)\}\big) = g\big(F_a\big(P; \{L_i^1\}, \{L_i^0\}\big)\big).$$

2.4. Hardy method

We now consider another class of interpolation methods: the radial basis function methods. The interpolation function is defined as the linear combination of radial basis functions $\mathbf{B}_i(\,)$, i.e.,

$$F(P) = \sum_{i=1}^{n} c_i \mathbf{B}_i(P). \tag{8}$$

One of the most well known and effective radial basis function is the Hardy multiquadrics, by R.L. Hardy in (1971). It is defined as:

$$\mathbf{B}_i(P) = \left(d_i^2(P) + r_i^2\right)^{\alpha},$$

where d_i is the distance function from L_i^1, r_i is the stiffness radius controlling the stiffness of the deformation around L_i^1, and $\alpha > 0$ is an exponent parameter. Thus, the Hardy interpolation function is:

$$F_h(P) = \sum_{i=1}^{n} h_i \cdot \left(d_i^2(P) + r_i^2\right)^{\alpha} + \mathcal{L}(P), \tag{9}$$

where $\mathcal{L}(P) = h_{n+1} \cdot x + h_{n+2} \cdot y + h_{n+3} \cdot z + h_{n+4}$ is a global linear term ensuring linear precision of the transformation, $\{h_i\}$ ($h_i \in R^3$, $i = 1, \ldots, n+4$) are called Hardy coefficients. The parameters $\{r_i\}$ can be arbitrary. But for smooth results, they are normally set to:

$$r_i = \min_{j \neq i} d_i(L_j^1)$$

to ensure that the morphing is soft in the area where $\{L_i^1\}$ are sparse, and stiff when $\{L_i^1\}$ are dense. The exponent parameter α is often set to either 0.5 or -1.

Computation of the Hardy coefficients is quite straightforward. If we substitute condition $F_h(L_j^1) = L_j^0$ ($j = 1, \ldots, n$) into Eq. (9), we get the following linear system of n equations:

$$\sum_{i=1}^{n} h_i \cdot \left(d_i^2(L_j^1) + r_i^2\right)^{\alpha} + \mathcal{L}(L_j^1) = L_j^0 \quad (j = 1, \ldots, n). \tag{10}$$

For linear precision, we also require:

$$\sum_{i=1}^{n} h_i \cdot x_i = \sum_{i=1}^{n} h_i \cdot y_i = \sum_{i=1}^{n} h_i \cdot z_i = \sum_{i=1}^{n} h_i = 0. \tag{11}$$

Together it is a linear system of $n+4$ equations with $n+4$ unknowns $\{h_i\}$ ($i = 1, \ldots, n+4$). As shown in Micchelli's paper (1986), this linear system is very stable, and the interpolant exists and is unique with normal datasets.

2.5. Locally Bounded Hardy method

Computing $F_h(P)$ is quite expensive because the distances from each point P to all landmarks need to be individually calculated, and there are usually a large number of points to be morphed in a volume. It was pointed out in (Franke and Nielson, 1980; Ruprecht and Muller, 1995) that, when $\alpha = -1$, an influence radius R_i can be defined for each landmark point L_i, so that its contribution (the ith term in the summation in (9)) to points outside its influence area (the sphere centered at L_i with radius R_i) is smaller than a pre-defined tolerance ε, and therefore can be neglected. As first described in (Franke and Nielson, 1980), this leads to the following locally bounded Hardy's interpolation:

$$F_b(P) = \sum_{i=1}^{n} H_i \cdot \left[\left(d_i^2(P) + r_i^2\right)^{-1} - \left(R_i^2 + r_i^2\right)^{-1}\right]_+ + \mathcal{L}(P), \tag{12}$$

where R_i is computed from the equation

$$|h_i| \cdot \left(R_i^2 + r_i^2\right)^{-1} = \varepsilon. \tag{13}$$

In function F_b, the ith term in the summation vanishes if the distance to L_i is greater than R_i. The tolerance ε determines the accuracy of the approximation of (12) to (9). To ensure that the influence area of each landmark contains at least one other landmark, we also require that $R_i > r_i$. To compute the coefficients $\{H_i\}$, we need to first solve the linear system (9) and then compute the influence radii $\{R_i\}$ from (13). Finally the following interpolation equations will be solved to obtain $\{H_i\}$:

$$F_b(L_i) = L_i^0. \tag{14}$$

Within the given error tolerance, $F_b(P)$ can be computed much faster than $F_h(P)$ because points outside the influence areas of the landmarks need not be considered. For the examples we have tested, an average speed up of more than ten can be achieved with ε being the size of one voxel.

2.6. Thin-Plate Spline method

Another well known radial basis function is the Thin-Plate Spline basis function. 2D Thin-Plate Spline has been successfully used in surface interpolation and deformation (Bookstein, 1989; Meinguet, 1979). Because of its property of minimum bending energy (Bookstein, 1989), A 3D extension of the Thin-Plate Spline function is a natural candidate for volume morphing.

A 3D Thin-Plate Spline can be defined as:

$$F_t = \sum_{i=1}^{n} w_i \mathbf{U}\left(d_i(P)\right) + \mathcal{L}(P),$$

where $\mathbf{U}(d_i(P)) = d_i^2(P)\log(d_i^2(P))$, $\mathcal{L}(P)$ serves the same purpose as in (9), and $w_i \in R^3$ are 3D spline coefficients that can be similarly computed as in the Hardy method.

Interestingly, when applied to the morphing between different species, F_t tends to generate drastic distortions in areas where landmarks are sparse. However, if we redefine

Table 1
Comparison table for different volume morphing methods

Method	Timing	Accuracy on LM recovery	Root mean square error	Affine invariance	Visual smoothness
Shepard ($r = 4$)	672	19.64	50.08	No	Poor
Locally Affine Shepard ($r = 4$)	1266	22.63	42.82	Yes	Good
Hardy ($\alpha = 0.5$)	721	20.91	20.58	Yes	Good
Locally Bounded Hardy ($\varepsilon = 1$)	55.8	21.13	46.75	Yes	Fair
Thin-Plate Spline	1384	19.47	23.95	Yes	Good

the basis function to $\mathbf{U}(d_i(P)) = d_i(P)\log(d_i(P))$, the results look much more pleasant—the deformations in these areas are generally smooth without extra distortions. Figs. 1(g) and 2(g) are generated by this modified method.

2.7. Comparisons

To compare the above five volume morphing methods, an experiment has been done using the morphing from a human head to a macaque head with 56 landmarks. The resulting images are shown in Fig. 1, and some performance data is given in Table 1.

Because of the large number of points to be computed in a volume morphing process, the function evaluation speed of the morphing is a major concern. In general, all the above methods, except the Locally Bounded Hardy method, are very time-consuming. The timing shown in Table 1 is the total time (in second) taken to morph all 256^3 voxels in a 256^3 volume. An SGI Indigo-2 High-Impact R10000 workstation was used in this experiment. Note that the timing does not include the time for volume construction and trilinear interpolation. It is clear from this comparison that the Locally Bounded Hardy method is by far the fastest among this group. Even so, after including the time for volume construction and trilinear interpolation, it still needs several minutes to construct a 256^3 volume. The other methods need significantly more time.

Affine invariance is important since the source and target landmark sets may be placed in totally different coordinate systems. Among the five methods, only the Shepard method does not possess this property. In fact, higher order polynomial precisions can also be achieved by using higher degree polynomial terms, instead of the global linear terms used in the three radial basis function methods or the local affine terms used in the Locally Affine Shepard method.

Two different accuracy measures, *Accuracy on landmark recovery* and *Root mean square error* are designed to provide some quantitative comparisons. *Accuracy on landmark*

recovery measures the approximation error of the interpolation on landmarks that are taken out of the interpolation scheme. The procedure is as follows: For $i = 1, \ldots, n$, (a) removing the ith landmarks, L_i^0 and L_i^1, from both the source and the target landmark sets; (b) generating the morphing function, F_i, by interpolating the rest of the landmarks; and (c) computing the error $\delta_i = \|F_i(L_i^1) - L_i^0\|$. The average error $\delta = \sum_{i=1}^{n} \delta_i / n$ is then taken as the accuracy measure on landmark recovery for a given interpolation scheme as shown in Table 1. The results show very small differences among the five interpolation methods.

Root mean square error measures the accuracy in reproducing a known function over a $N \times M \times L$ volume. The error is defined as:

$$E_f(F) = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{L} \frac{(f(i,j,k) - F(i,j,k))^2}{NML}},$$

where $F(\,)$ is one of the interpolation functions, and $f(\,)$ is a known function from which the landmarks are computed. For reference purpose, and to make the results less dependent on the test functions, we compute the average of the error estimates $E_f(F)$ with all six test functions, $f_1, \ldots, f_6$, originally defined and used in (Franke, 1982; Nielson, 1993) (for detailed definitions of the test functions, see (Franke, 1982; Nielson, 1993). Thus, for each morphing method with interpolation function F, we compute $E(F) = \sum_{i=1}^{6} E_{f_i}(F)/6$, and enter the result to Table 1 under the column of *Root mean square error*. In our experiment, the 56 landmarks placed in the Macaque volume are used as the target landmarks $\{L_i\}$, and the source landmarks are directly computed by $L_i^0 = f(L_i)$ for each test function f. The error term E is calculated over all voxels in a 256^3 volume. The large errors shown in Table 1 are as expected since 56 landmarks in a 256^3 volume is hardly sufficient for approximation purpose. Nevertheless, the results in Table 1 show that the Hardy method and the Thin-Plate Spline method have a better overall accuracy.

Although the *Root mean square error* provides some level of error estimates of the different interpolation methods, it does not convey much information about their visual appearance and smoothness. *Visual smoothness* refers to the general visual pleasantness of the resulting shapes. It is a very subjective criterion. Nevertheless, from our observation, the Hardy method, the Thin-Plate Spline method and the Locally Affine method all seem to generate good morphing results. The Locally Bounded Hardy method generates fair results, but it is a lot faster than all other methods. The Shepard method sometime creates unpleasant artifacts. This is probably due to the fact that the first derivatives of the Shepard morphing function are always zero at landmark points, resulting in a flat spot around each landmark point. To provide more visual information for assessing *Visual Smoothness*, we also applied the five interpolation methods to a young child skull with target landmarks defined on a skull of older age. Although the two skulls are not from the same person, they are both considered typical normal skulls. Thus it is reasonable to expect that the actual growth result of the child skull should be fairly close to the older skull if their landmarks match (Note that since the morphing only uses the volume data of the younger skull, the growth results will keep all features of the younger skull, with its shape changed). The results given in Fig. 2 show again that the Hardy method, the Thin-Plate Spline method and the Locally Affine method generate better (in terms of *Visual Smoothness*) morphing results.

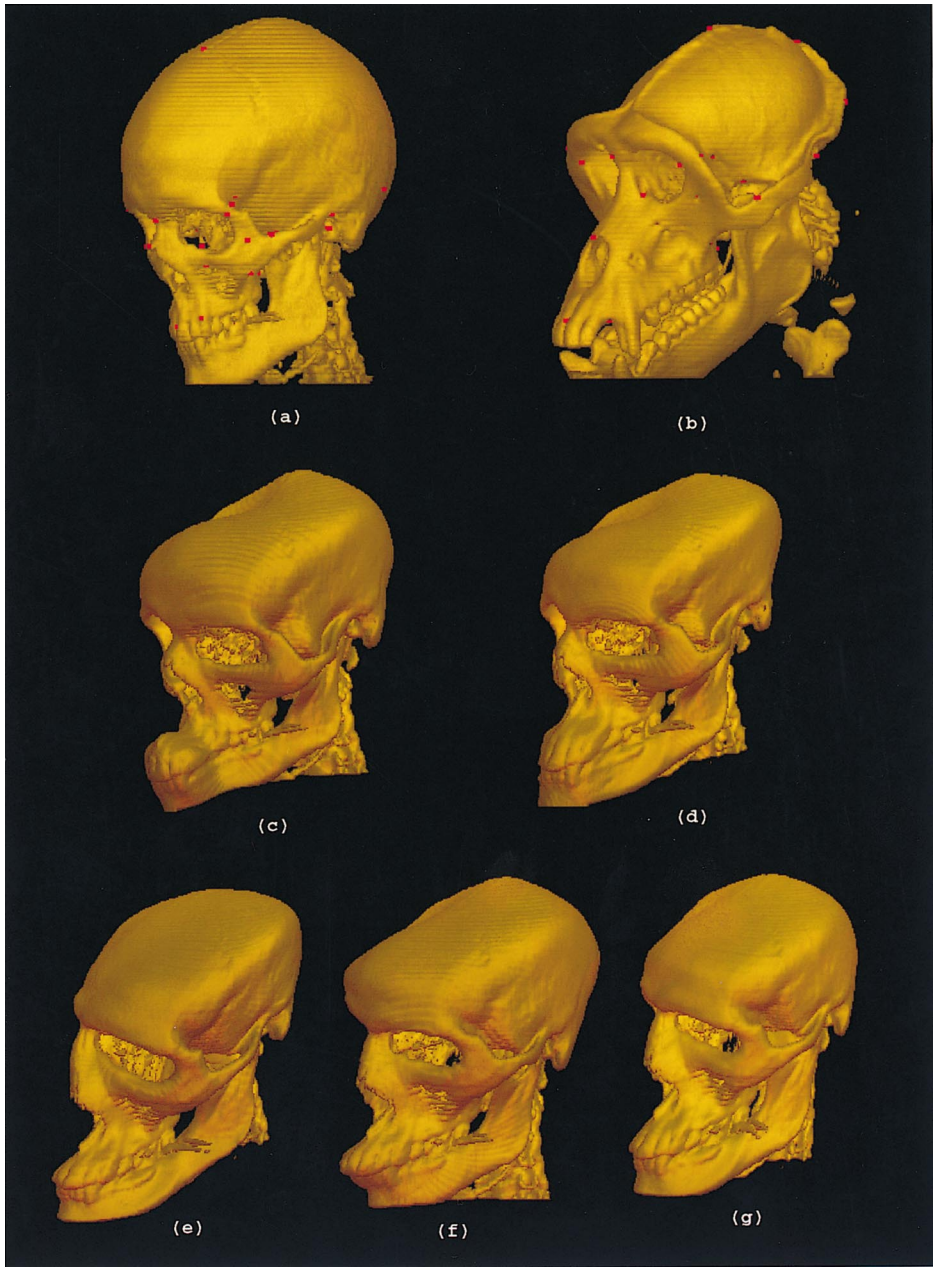

Fig. 1. Morphing a human head to a macaque's landmarks: (a) original human head with landmarks; (b) original Macaque head with landmarks; (c) Shepard method; (d) Locally Affine Shepard method; (e) Hardy method; (f) Locally Bounded Hardy method; (g) Thin-Plate Spline method.

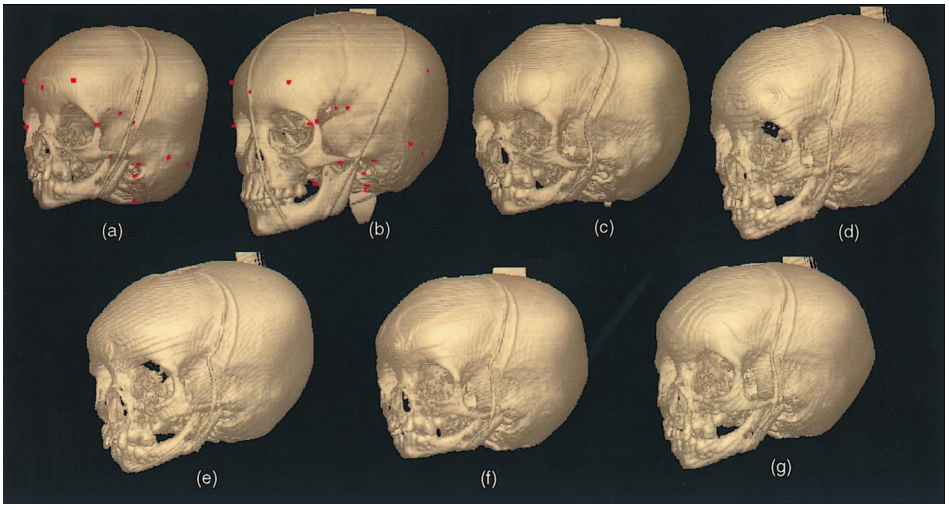

Fig. 2. Morphing a young child skull to the landmarks of an older skull: (a) young child skull with landmarks; (b) older skull with landmarks; (c) Shepard method; (d) Locally Affine Shepard method; (e) Hardy method; (f) Locally Bounded Hardy method; (g) Thin-Plate Spline method.

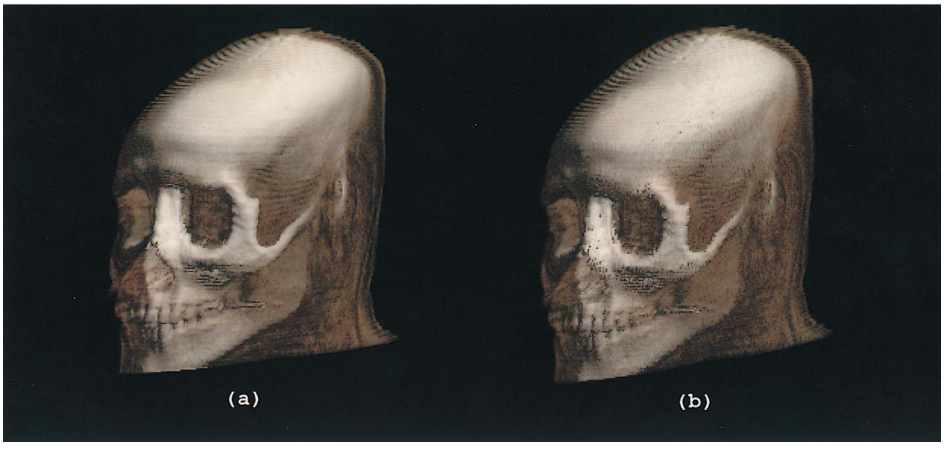

Fig. 4. Morphing from human to cebus: (a) rendered directly from the morphing function, 152 seconds; (b) rendered with approximation of the morphing function, 24 seconds.

3. Deformable volume rendering

A backward deformation function can be obtained from one of the methods described in the last section. In order to render the volume deformation process, the conventional approach is to construct an intermediate deformed volume for each frame of the deformation. As shown in Section 2.7, this is too expensive for many practical applications.

In this section, we will develop an integrated volume rendering algorithm that takes only the original volume with a deformation function, and directly renders the deformed volumes without the intermediate volume construction. The algorithm uses raycasting formulation, and is optimized in speed and memory for parallel projection renderings, as shown in Table 2. Faster speed can be achieved by using hardware assisted 3D texture mapping (Fang et al., 1996b; Westermann and Thomas, 1998), with a price of lower image quality and higher hardware cost.

3.1. Adaptive subdivision for target blocks

The first step of the algorithm is to adaptively subdivide the target volume space into target blocks for the approximation of the deformation function F, i.e., within each target block, only the block vertices need to be computed by F, and all other points can be trilinearly interpolated. To efficiently represent such adaptive subdivision process, an octree data structure is utilized. This target octree is initially set to a complete tree representing a preliminary uniform subdivision with a pre-determined depth level.

At any point of the subdivision, the current leaf nodes of the octree are considered the current target blocks, and will then be recursively subdivided into eight smaller target blocks until the function F can be approximated by trilinear interpolation within the blocks. To determine whether a block needs to be subdivided, we adopt an approach used in (Lerios et al., 1995). In this approach, several sample points in each target block are selected (often the center of the block and the centers of its six faces, since these points will be part of the vertices if subdivision is needed, and will then need to be computed anyway) and deformed separately by the morphing function F as well as the trilinear interpolation. If the distances between the morphed results and the interpolated results are under the given tolerance, the block is considered acceptable, otherwise subdivision is needed. This process continues recursively until all target blocks meet the tolerance requirement. Apparently, this heuristic is not mathematically accurate, but it is easy to use and works in most cases when the morphing is not too violent. A more theoretically sound approach would be to compute all the local extrema of the morphing function to measure how violent the morphing is in a given area, and then determine appropriate tolerance values for different areas. But this would be too expensive, and may unnecessarily generate a lot more blocks.

The above subdivision approach, however, is likely to create sampling discontinuity between the neighboring blocks that are created at different levels of subdivision. As shown in Fig. 3, a small gap (or overlap) may occur along the common boundary of the neighboring blocks as a result of linear interpolation. Consequently, the volume data within this small gap or overlapping area can be mis-sampled (omitted) or over-sampled (stretched). This may lead to some feature getting slightly thicker or thinner in the final image, and it may also generate small artifacts due to the potential uneven sampling from such gaps. But since mis-sampling or over-sampling does not occur in the target volume, it will not generate cracks in the final image. A comparison example is given in Fig. 4 with timings. Fig. 4(a) is rendered without any approximation, i.e., all sampling points are morphed directly through the morphing function. Fig. 4(b) is rendered using the

Table 2
Rendering performance data

Dataset	Rendering time (sec)				Morphing & tgt blk setup time (sec)	# of tgt blks	Memory (MB)
	Build template	Block projection	Resam., blending & shading	Total			
H-M-128-T	0.002	1.81 (19.71)	9.288	11.1	5.8	2093	1.34
H-M-256-T	0.004	3.24 (32.47)	31.036	34.28	6.3	2141	9.95
H-C-128-T	0.002	1.74 (18.08)	8.308	10.05	5.4	1999	1.30
H-C-256-T	0.007	3.14 (31.3)	34.913	38.06	5.5	2026	9.87
E-H-256-Sk	0.003	3.25 (36.69)	11.497	14.75	3.96	1508	3.37
H-Ch-256-Sk	0.007	4.18 (35.25)	24.533	28.72	6.0	2283	9.89
H-E-256-Sk	0.008	3.12 (36.01)	18.072	21.20	4.16	2278	9.81
H-S-256-T	0.006	5.9 (52.4)	25.994	31.9	5.18	1947	9.92

Keys: H-human, M-macaque, C-cebus, E-erectus, Ch-chimpanzee, S-saimiri, T-translucent, Sk-skull, sk-skin.
Shading parameters: 5 light sources, Zucker-Hummel based gradient estimation, trilinear interpolation for resampling, depth cueing for non-translucent rendering.
Timings in parentheses in “block projection” category are measured without the template-based optimization.

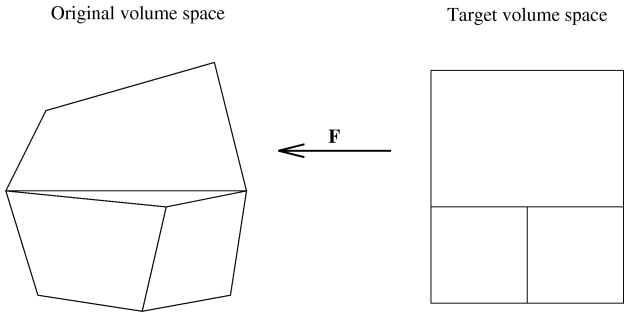

Fig. 3. Discontinuity in neighboring blocks with different levels of subdivisions.

Z	Y	X	**Front-to-back order**
<0	<0	<0	7, 6, 5, 3, 4, 2, 1, 0
<0	<0	>=0	6, 7, 4, 2, 5, 3, 0, 1
<0	>=0	<0	5, 4, 7, 1, 6, 0, 3, 2
<0	>=0	>=0	4, 5, 6, 0, 7, 1, 2, 3
>=0	<0	<0	3, 2, 1, 7, 0, 6, 5, 4
>=0	<0	>=0	2, 3, 0, 6, 1, 7, 4, 5
>=0	>=0	<0	1, 0, 3, 5, 2, 4, 7, 6
>=0	>=0	>=0	0, 1, 2, 4, 3, 5, 6, 7

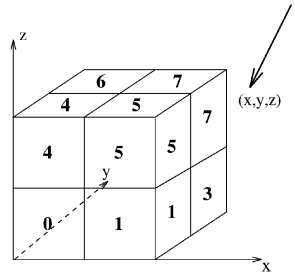

Fig. 5. The order of the octants for a given viewing direction.

approximation described here with a tolerance of the size of three voxels. Although image quality may suffer a little due to the approximation, the significant saving in computation time still makes this approach attractive.

3.2. Sorting of octree blocks

The next step in the rendering algorithm is to sort the target blocks created from the subdivision process in a front-to-back order so that the blocks can be projected according to their viewing order.

Since the target blocks are already organized in the target octree, sorting for a given viewing direction is fairly simple. Assuming the viewing direction vector is (x, y, z), relative to the orthogonal coordinate system of the octree, the signs of the coordinates determine the order in which the eight child octants of each node are visited, as shown in Fig. 5. Based on this view-dependent child access order, a view-dependent depth-first traversal of the target octree will automatically pick the target blocks in front-to-back order.

3.3. Target block projection

The final step of the rendering algorithm is to project all the target blocks to the projection plane in a front-to-back order. This projection is essentially a raycasting process within the block, except that the voxel values are not directly available from the target block—they have to be fetched from the original volume through the deformation function. Each of such block projections consists of three steps:

(1) Computing the intersection between the rays emanating from all pixels and the block. The results are all the ray segments that reside within the target block.
(2) Sampling each intersecting ray segment and performing color/opacity compositing and blending. This involves the morphing of the sampling points to obtain their color/opacity values from the original volume.
(3) Shading of the sampling points based on the target volume's gradients at the sampling points.

Taking advantages of the uniform block orientation and sizes, we developed a template-based approach that provides very effective optimization to the block projection process. It should be noted that the template-based approach described here only works for parallel projections. Block coherence for perspective projections is much more difficult to define. Nevertheless the overall rendering algorithm will still work for perspective projections without the template, though it will be significantly slower.

3.3.1. Template-based ray-block intersection

Since there can be hundreds and sometimes thousands of target blocks, computing the ray-block intersection for all of them directly can be quite expensive. For parallel projection, however, a template can be used to take advantage of the fact that all target blocks have the same orientation, shape and only a few fixed sizes. Intuitively, the result of ray-block intersection for one sample should be usable by all other blocks of the same size with a simple translation. To describe this approach, let's first define

(1) the *active pixels*: the pixels that the block projects to;
(2) the *active sampling points*: the raycasting sampling points along the rays from the *active pixels* that are within the block.

For blocks of the same size, the *active pixels* and *active sampling points* should be identified only once, saved in a template, and pasted to all other blocks.

A template is defined as a 2D array containing the screen area of the *active pixels*. Each entry of the 2D array contains simply two index numbers indicating the interval of the *active sampling points*. To build the template, a sample block for each block size is used. The 2D screen projection of the bounding box of the rotated sample block forms a rectangular screen area containing all the *active pixels* of the sample block. For every ray emanating from each *active pixel*, the ray segment that is within the block is computed by normal ray-block intersection, and the interval of the sampling points on this ray segment is stored in the template.

For each block in the sorted block list, a displacement vector from the sample block is first computed. A template of appropriate size is translated by the displacement vector to obtain the *active pixels* and *active sampling points* of the block.

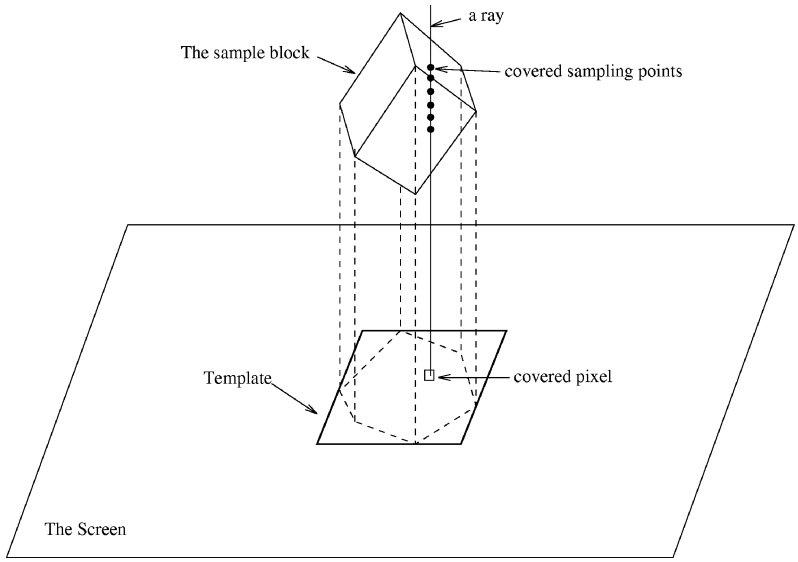

Fig. 6. The template for block projection.

There is one complication, however, in using the template. The set of all raycasting sampling points forms a 3D grid in the viewing space. If we use the center of the rotated block as the reference point, and define the *block offset* as the offset of the reference point to its nearest grid point, it is easy to see that such offsets are usually different for different blocks, even if they have the same size and orientation. Ideally, if the offset of the sample block used for building the template is the same as that of the block to be projected, the template translation would be straightforward. But when their offsets are different, the *active pixels* and *active sampling points* obtained from the translated template can be slightly different.

Our solution to this problem is to slightly increase the size of the sample block to form a so called *extended sample block* to build the template to ensure that all possible offsets are covered. Without loss of generality, we assume that the resolution (grid step) of the sampling grid is 1. Thus, the offset values in X, Y and Z for any block can only be between -0.5 and 0.5. In other words, an extended sample block with reference point at a grid point P should contain the union of all the possible sample blocks of the given size with their reference points within a unit cubic cell centered at P. To illustrate this idea, a 2D analog is shown in Fig. 7. The extended sample block is obtained by extending the four (six in 3D) sides of the original sample block (dashed) outward by an amount of $\sqrt{2}$ (the largest offset magnitude, should be $\sqrt{3}$ in 3D). This is to ensure that all sample blocks with reference points within the unit cell are included. As shown in Fig. 7, the sample block with its reference point Q at a corner of the unit cell, for instance, is still contained within the extended sample block.

This solution only ensures that the set of *active pixels* and *active sampling points* of a block is a subset of the translated template. The extra sampling points from the translated

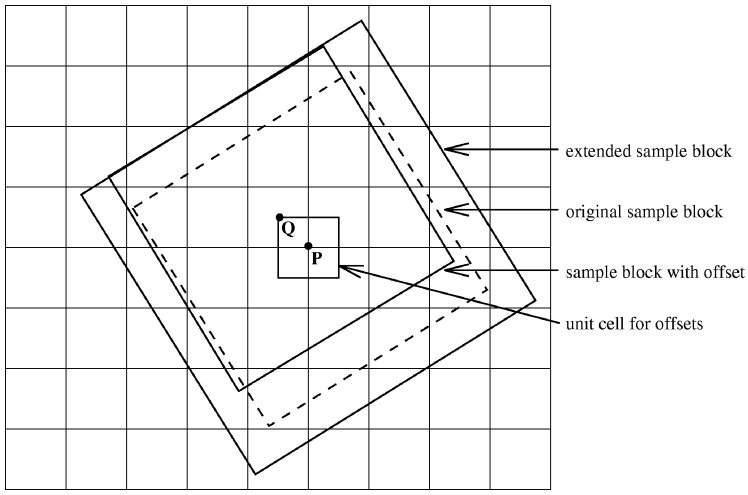

Fig. 7. A 2D analog of the extended sample block.

template, due to the enlarged sample block, need to be removed to avoid double sampling. To do so, during the projection of a block, we have to first test a few points at the start and end of each ray segment to remove those that are not within the block. Fortunately, such test is not expensive since, in general, only one or two points need to be skipped, thus the overhead of this test is fairly small compared to the savings.

3.3.2. Morphing of the sampling points

In order to perform color/opacity compositing and blending, the color/opacity values of the sampling points on the intersecting ray segments have to be obtained from their morphed points in the source volume. As shown in Section 2.7, computing the morphing function for all the sampling points is costly. However, since the morphing can be approximated by trilinear interpolation within each target block, only the eight vertices of each target block need to be morphed, and all other sampling points in the block can be interpolated from these eight vertices. This greatly reduces the morphing computation time, and allows for faster overall rendering speed. Animation of the intermediate steps of the morphing can be simply done by linear interpolation of the original block vertices and their morphed results.

3.3.3. Gradient computation for shading

Shading is done the same way as normal volume rendering (Levoy, 1988) except that the gradient vector needed for the shading formula is not directly available from the target volume, since the voxel values are not explicitly stored in the target volume. It can, however, be computed through the original volume and the morphing function F.

Let the intensity function of the source volume V_0 be $f : R^3 \to R$, the intensity function of the target volume V_1 be $g : R^3 \to R$. For any point $Q = (u, v, w) \in V_1$, its morphed point is $P = (x, y, z) = F(Q) \in V_0$, i.e., $g(Q) = f(P) = f(F(Q))$.

The gradient of V_1, $D_g = (\frac{\partial g}{\partial u}, \frac{\partial g}{\partial v}, \frac{\partial g}{\partial w})^{\mathrm{T}}$, can be computed as follows:

$$D_g(Q) = J_F(Q) \cdot D_f(P),$$

where $D_f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})^{\mathrm{T}}$ is the gradient of the source volume, and J_F is the Jacobian matrix of the morphing function:

$$J_F = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \\ \frac{\partial x}{\partial w} & \frac{\partial y}{\partial w} & \frac{\partial z}{\partial w} \end{pmatrix} = \left(\frac{\partial F}{\partial u}, \quad \frac{\partial F}{\partial v}, \quad \frac{\partial F}{\partial w} \right)^{\mathrm{T}}.$$

While D_f is directly available from V_0, computing J_F for every point is, however, expensive. But since within each target block the function F can be approximated by trilinear interpolation, similar to the morphing function evaluation, we only need to compute the Jacobian matrices for the block vertices and use trilinear interpolation to get the Jacobian matrices for all other points within the block.

3.4. Implementation

The algorithm presented here has been implemented in C++ on an SGI-VTX (R4400) workstation of 150 MHz with 128 Mb memory. Some examples are given in Fig. 8. To achieve the maximum speed, we used the locally bounded Hardy method as the deformation function, with $\varepsilon = 1$. A tolerance of the size of one voxel was also used in the adaptive subdivision process for better image quality. In Fig. 8, CT-scanned data sets of a modern human (*Homo sapiens sapiens*) head, some primate heads, and casts of fossil primates were used. Approximately 56 homologous biological landmark points were located on these datasets at the Johns Hopkins University School of Medicine. Fig. 8(a) is the original modern human head rendered with translucency. Figs. 8(b) and (c) are the morphed results (translucent using landmarks collected from an Old World monkey (Macaque mulatta) for 256^3 and 128^3 volumes, respectively. Fig. 8(d) is a rendered skin images of a 256^3 volume morphing from modern human to the fossil cast of *Homo erectus*. Fig. 8(e) shows an image of *Homo erectus* from a 256^3 volume, which is morphed to the modern human configuration (Fig. 8(f)). Figs. 8(g)–(i) show some more deformable rendering results for morphed 256^3 volumes.

The algorithm's performance data for the images in Fig. 8 are given in Table 2. A partitioned rendering time is given for each case. It is the rendering time for each new angle or deformation frame. The setup time is for building the morphing function and the target blocks, and occurs only once when there is a change of landmarks. To demonstrate the savings from the template-based optimization, a separate timing is measured without the template (i.e., the ray-block intersections are computed individually for all blocks) for each case, and shown (in parentheses) in the *block projection* category in Table 2. In the table, the total memory usage is also given. It includes the representation of the original volume, the target octree and templates. Since an octree volume representation is used to store the original volume as well, the total memory use is fairly small.

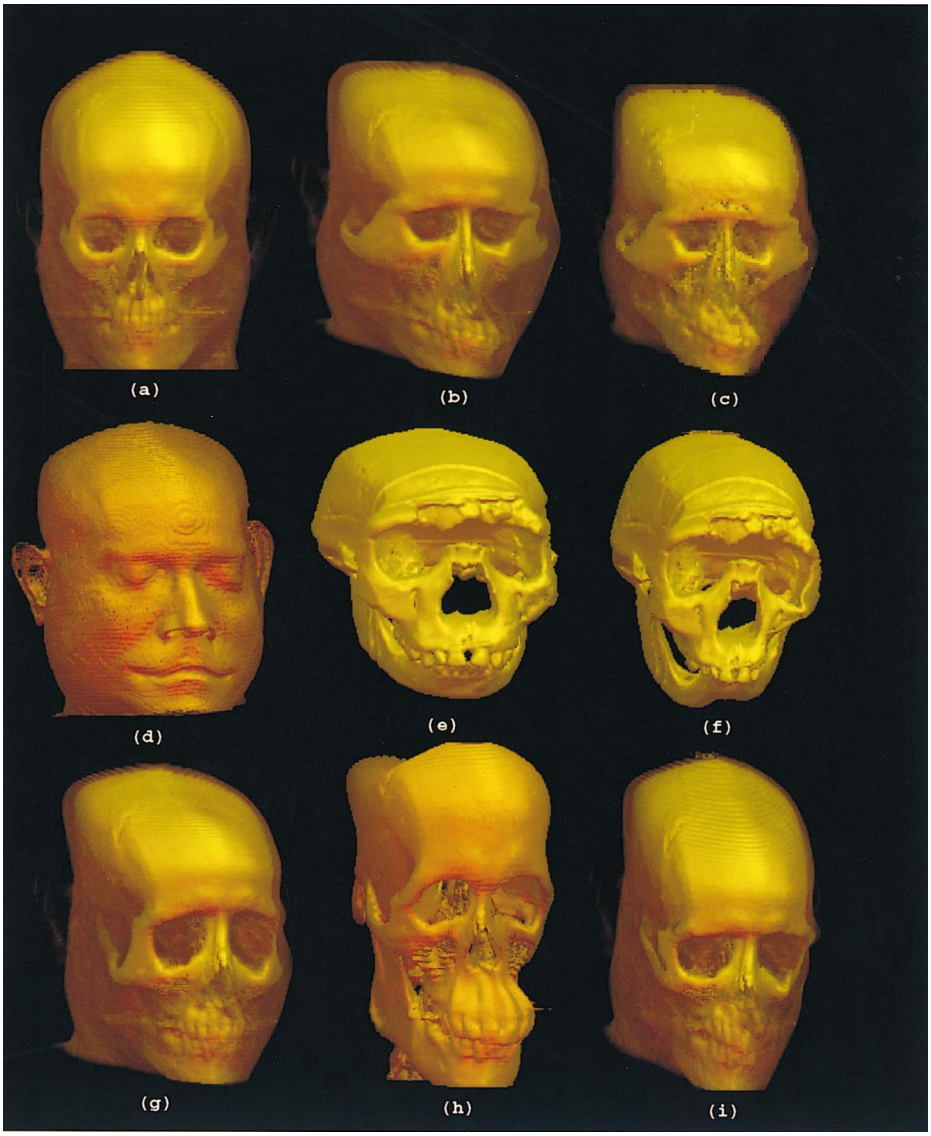

Fig. 8. (a) Human (256^3); (b) Human-Macaque (256^3); (c) Human-Macaque (128^3); (d) Human-Erectus (256^3); (e) Erectus (256^3); (f) Erectus-Human (256^3); (g) Human-Cebus (256^3); (h) Human-Chimpanzee (256^3); (i) Human-Saimiri (256^3).

In a separate application, we used this algorithm to simulate craniofacial patient's growth process. In Fig. 9, a post-operative patient's CT volume (Fig. 9(a)) is grown to seven years old using an abnormal (sagittal synostosis) growth pattern (Fig. 9(b)). Fig. 9(c) shows the same growth to thirty-seven years old.

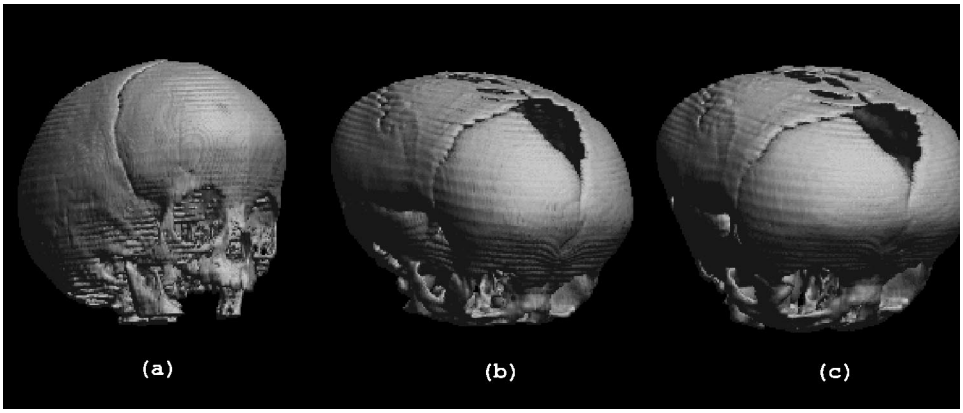

Fig. 9. (a) The post-operative image (256^3); (b) Growth to 7 years old with sagittal pattern (256^3); (c) Growth to 37 years old with sagittal pattern (256^3).

4. Discussions and conclusions

In this paper, we presented a new approach to the integration of volume morphing and volume rendering. Various landmark interpolation methods are discussed and applied for the constructions of volume morphing functions. An object space deformable volume rendering algorithm is developed to directly render the deformed volume without constructing intermediate deformed volumes. Comparing to previous work by Lerios et al. (1995), and by Kurzion and Yagel (1997), our approach has the following two major advantages:

(1) Our landmark-based morphing is a more useful tool for deformation modeling. Since the movements of the landmarks can often be computed through a deformation modeling system, such as a physically-based modeling system, a particle system or a biological model, the volume morphing by interpolating such landmark movements becomes an approximated physical or biological models for a real deformation problem. This is particularly true for biomedical applications where almost all deformation and growth studies are focused on biological point landmarks only (Bookstein, 1991). One such application is the post-operative growth prediction, with a known landmark growth model (Richtsmeier and Lele, 1993), in craniofacial surgeries for children with craniofacial anomalies (Dufresne et al., 1995; Richtsmeier et al., 1997). For deformation modeling purpose, morphing using interactive geometric operations such as the ray deflectors in (Kurzion and Yagel, 1997) are not very useful. In addition, higher dimensional feature landmarks, such as lines and boxes used in (Lerios et al., 1995), are generally not available either.

(2) Our approach integrates the two separate processes of morphing and rendering in (Lerios et al., 1995), and is more efficient for two reasons. First, the integrated approach only need to compute the deformation for visible voxels only, while the volume reconstruction process has to compute the deformation of all voxels since no visibility information is available in this step. This is particularly important when

the objects are relatively opaque and the deformation and its rendering are dynamic. Secondly, since both volume reconstruction and rendering involve resampling of the volume dataset, the volume reconstruction based approach (Lerios et al., 1995) leads to repeated resampling, cumulative resampling errors and unnecessary interpolation computation, while our approach integrates all the resampling into one in the final rendering step only.

In the future, we will further explore the potential integration of landmark movement modeling, volume morphing and rendering to create a truly volume deformation modeling environment. We will also study the block coherence for perspective viewing to extend the template-based rendering technique to perspective projections. Further improvement in rendering speed is still very desirable for interactive volume deformation and visualization.

Acknowledgements

We would like to thank Su Huang for implementing the octree volume representation and operations, and Meiyappan Solaiyappan for his early work in using Shepard method in morphing and the subsequent texture mapping rendering. The inspiration of this work comes from the original CI$_{\text{E}}$MED project, *Mutability*. We would like to take this opportunity to thank all the other members of the *Mutability* project, in particular, Dr. Tim Poston, Dr. Wayne Lawton and H.T. Nguyen. We would also like to thank the reviewers for their detailed reviews and excellent suggestions for the improvement of this paper.

References

Arad, N., Dyn, N., Reisfeld, D. and Yeshurun, Y. (1994), Image warping by radial basis functions: Application to facial expressions, Computer Vision, Graphics, and Image Processing 56 (2), 161–172.

Beier, T. and Neely, S. (1992), Feature-based image metamorphosis, Computer Graphics, SIGGRAPH'92 (26) 2, 35–42.

Bookstein, F.L. (1989), Principal warps: Thin-plate splines and the decomposition of deformations, IEEE Trans. Pattern Analysis and Machine Intelligence 11 (6), 567–585.

Bookstein, F.L. (1991), Morphometric Tools for Landmark Data: Geometry and Biology, Cambridge University Press.

Danskin, J. and Hanrahan, P. (1992), Fast algorithms for volume ray tracing, in: Proc. 1992 Workshop on Volume Visualization, 91–98.

Dufresne, C., Raghavan, R., Fang, S. and Richtsmeier, J. (1995), Computerized dynamic skeletal modeling for craniofacial surgical planning: New tools to predict growth following surgery, in: Proc. VIth International Congress of the International Society of Craniofacial Surgery, Saint Tropez, France, 31–32.

Fang, S., Raghavan, R. and Richtsmeier, J.T. (1996a), Volume morphing methods for landmark-based 3D image deformation, in: Proc. 1996 SPIE Medical Imaging, SPIE 2710, Newport Beach, CA, 404–415.

Fang, S., Srinivasan, R., Huang, S. and Raghavan, R. (1996b), Deformable volume rendering by 3D texture mapping and octree encoding, in: Proc. IEEE Visualization'96, San Francisco, CA, 73–80.

Franke, R. (1982), Scattered data interpolation: tests of some methods, Math. Comp. 38, 181–200.

Franke, R. and Nielson, G. (1980), Smooth interpolation of large sets of scattered data, Int. J. Numerical Methods in Engineering 15, 1691–1704.
Hardy, R.L. (1971), Multiquadric equations of topography and other irregular surfaces, J. Geophys. Res. 76, 1905–1915.
He, T., Wang, S. and Kaufman, A. (1994), Wavelet-based volume morphing, in: IEEE Visualization 94, 85–91.
Hughes, J.F. (1992), Scheduled Fourier volume morphing, Computer Graphics, SIGGRAPH'92 26 (2), 43–46.
Koyamada, K. and Ito, T. (1995), Fast generation of spherical slicing surfaces for irregular volume rendering, The Visual Computer 11 (3), 167–176.
Kurzion, Y. and Yagel, R. (1997), Interactive space deformation with hardware assisted rendering, IEEE Computer Graphics and Application 17.
Laur, D. and Hanrahan, P. (1991), Hierarchical splatting: A progressive refinement algorithm for volume rendering, Computer Graphics, SIGGRAPH'91 (25) 4, 285–288.
Lee, S.Y., Chwa, K.Y. and Shin, S.Y. (1995), Image metamorphosis using snakes and free-form deformations, SIGGRAPH'95, 439–448.
Lerios, A., Garfinkle, C.D. and Levoy, M. (1995), Feature-based volume metamorphosis, SIGGRAPH'95, 449–456.
Levoy, M. (1988), Display of surfaces from volume data, IEEE Computer Graphics and Application 8 (3), 29–37.
Levoy, M. (1990), Efficient ray tracing of volume data, ACM Trans. on Graphics 9 (3), 245–261.
Meinguet, J. (1979), Multivariate interpolation at arbitrary points made simple, Zeitschrift für Angewandte Mathematik und Physik (ZAMP) 30, 292–304.
Micchelli, C.A. (1986), Interpolation of scattered data: Distance matrices and conditionally positive definite functions, Constr. Approx. 2, 11–22.
Nielson, G.M. (1993), Scattered data modeling, IEEE Computer Graphics and Application 13 (1), 60–70.
Richtsmeier, J.T., Fang, S., Solaiyappan, M. and Raghavan, R. (1997), Volumetric morphing, the study of craniofacial growth and the prediction of surgical outcome, Cleft Palate-Craniofacial Journal, in revision.
Richtsmeier, J.T. and Lele, S. (1993), A coordinate-free approach to the analysis of growth patterns: Models and theoretical considerations, Biol. Review 68, 381–411.
Ruprecht, D. and Muller, H. (1995), Image warping with scattered data interpolation, IEEE Computer Graphics and Application 15 (2), 37–43.
Shepard, D. (1968), A two-dimensional interpolation function for irregularly spaced data, in: Proc. 23rd National Conference of the ACM, New York, 517–524.
Silva, C., Mitchell, J.S.B. and Kaufman, A.E. (1996), Fast rendering of irregular grids, in: 1996 Volume Visualization Symposium, IEEE, 15–22.
Tost, D., Puig, A. and Navazo, I. (1995), A volume visualization algorithm using a coherent extended weight matrix, Computer and Graphics 19 (1), 37–45.
Westermann, R. and Thomas, E. (1998), Efficiently using graphics hardware in volume rendering applications, in: SIGGRAPH'98, 169–177.
Wilhelms, J. and Gelder, A.V. (1991), A coherent projection approach for direct volume rendering, Computer Graphics, SIGGRAPH'91 25 (4), 275–284.
Wilhelms, J.P., Van Gelder, A., Tarantino, P. and Gibbs, J. (1996), Hierarchical and parallelizable direct volume rendering for irregular and multiple grids, in: IEEE Visualization'96.
Wolberg, G. (1990), Digital Image Warping, IEEE Computer Society Press.
Yagel, R. and Kaufman, A. (1992), Template-based volume viewing, in: Eurographics'92, 153–167.