

# Methods for Volume Metamorphosis

M. Chen, M.W. Jones and P. Townsend

Department of Computer Science, University of Wales Swansea,  
Singleton Park, Swansea SA2 8PP, United Kingdom  
E-mail: {m.chen, m.w.jones, p.townsend}@swansea.ac.uk

## Abstract

The problem of creating a sequence of inbetween volumes smoothly transforming from one given volume to another is referred to as volume metamorphosis or volume morphing. Several methods that provide the solution to the problem are described and compared in this paper. Tests have shown that some of these methods are capable of produce good quality metamorphosis within an acceptable time.

## 1 Introduction

Volume data, which is used widely in scientific computation and medical imaging to represent 3D scalar fields, provides a uniform way of modelling 3D objects. As voxel calculations are much simpler than polygon calculations, most algorithms for manipulating and rendering volume data are suitable to be embedded into special hardware [1, 2]. For modelling 2D objects in video production, perhaps there is no doubt that image representations have significant advantages compared with vector representations. When it comes to the synthesis of 3D models for creating images sequences, volume-based techniques cannot be overlooked.

*Image metamorphosis* [3, 4], more commonly called image morphing, is a computer animation technique for creating an image sequence smoothly transforming from one given image to another. The technique has recently received much attention, especially from the film and television industry. Also relatively more recently, graphics tools have become widely available for visualising volume datasets [5], and such tools have the potential to be developed into methods for volume metamorphosis. In addition to the synthesis of 3D models in video production, the application areas of such methods include computer animation, scientific visualisation, medical imaging and many aspects of natural science.

In this paper, several methods for volume metamorphosis are presented, most of which are developed based on the 2D methods widely used for image metamorphosis. A classification of morphing algorithms is given, and a set of volume distortion techniques, many of which have not yet been documented in the literature, are described. In addition to the static analysis of the complexity and usability of these methods, some graphical results and data of their run-time performance are presented.

## 2 The Process of Volume Metamorphosis

A general volume is collection of scattered voxels, each of which is associated with a set of values (of size  $S$ ), that is

$$\mathbf{V} = \{(x_i, v_i) \mid x_i \in \mathcal{R}^3, v_i \in \mathcal{R}^S, i = 1, 2, \dots, n\}. \quad (1)$$

One of the most commonly used volume representations is one where voxels are organised in the form of a 3D regular grid and each voxel is associated with a value. We will use the general volume representation in the following discussions, with the assumption that all volumes are of the same size and there is a one-to-one correspondence between voxels in any two volumes.

Given a starting volume  $\mathbf{V}_a$  and a finishing volume  $\mathbf{V}_b$ , the metamorphosis between them is a process which generates a sequence of inbetween volumes,  $\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_L$ , obtained using an interpolation function

$$\mathbf{V}_k \leftarrow F_{interp}\left(\mathbf{V}_a, \mathbf{V}_b, \frac{k}{L+1}\right), [k = 1, 2, \dots, L]. \quad (2)$$

This sequence of volumes represents a smooth transformation from  $\mathbf{V}_a$  to  $\mathbf{V}_b$ , and may then be rendered to give an illusion of continuous metamorphosis. Perhaps it is arguable that such an illusion could also be obtained by morphing between two images resulted from rendering  $\mathbf{V}_a$  and  $\mathbf{V}_b$ , or by interpolating the isosurfaces [6] of  $\mathbf{V}_a$  and  $\mathbf{V}_b$ . However, it would be extremely difficult for the image-based morphing to accommodate 3D geometric changes between two volumes, while the isosurface interpolation would require the establishment of a complex correspondence between two surfaces. Neither of the methods would be able to retain complete volumetric information in the inbetween sequence.

The morphing process is usually controlled by two control datasets,  $\mathbf{C}_a$  and  $\mathbf{C}_b$ , which specify the key features in  $\mathbf{V}_a$  and  $\mathbf{V}_b$  and their correspondence. Prior to the interpolation,  $\mathbf{V}_a$  and  $\mathbf{V}_b$  are deformed according to  $\mathbf{C}_a$  and  $\mathbf{C}_b$  so that every pair of corresponding key features have the same location. This leads to the problem of volume distortion that can be formulated with a distortion function

$$\mathbf{V}'_a \leftarrow F_{dist}(\mathbf{V}_a, \mathbf{C}_a, \mathbf{C}_k), \quad \mathbf{V}'_b \leftarrow F_{dist}(\mathbf{V}_b, \mathbf{C}_b, \mathbf{C}_k); \quad (3)$$

where  $\mathbf{C}_k$  is obtained in the same manner as  $\mathbf{V}_k$  in (2). In some applications, volume distortion may be used beyond the context of volume metamorphosis.

## 3 Volume Morphing Methods

According to the use of control datasets, approaches to volume metamorphosis can be classified into three categories, namely

- *cross dissolving* — which requires no control datasets;
- *field morphing* — where control datasets are used to specify coordinate mappings; and
- *mesh warping* — where control datasets define volume subdivisions as well as coordinate mappings.

### 3.1 Cross Dissolving

Cross dissolving is a process of metamorphosis that involves the interpolation but not the distortion of volumes. The following pseudo-code outlines this process

---

```
Procedure:    VolumeMetamorphosis-CrossDissolving;  
Input:      Va, Vb: Volume; L: Integer;  
Output:     Vout[1], Vout[2], ..., Vout[L]: Volume;
```

---

```
for k:=1 to L do  
    t = k / (L + 1.0);  
    Vout[k] := InterpolateVolume(Va, Vb, t);  
end for;
```

---

The simplest cross dissolving method is a linear interpolation between  $\mathbf{V}_a$  and  $\mathbf{V}_b$ , which often yields unsatisfactory results. To enhance the smoothness of the inbetween volumes, Fourier transformation may be used to schedule the interpolation non-linearly by favouring voxels whose values are close to the threshold of an interested isosurface [7]. Although they are easy to use and fast to run, both methods have difficulty in producing high quality results in most cases, especially when the transformations between two volumes involve scaling or rotation.

### 3.2 Field morphing

Given a pair of volumes,  $\mathbf{V}_a$  and  $\mathbf{V}_b$ , and their associated control datasets,  $\mathbf{C}_a$  and  $\mathbf{C}_b$ , the following pseudo-code outlines the general process of field morphing.

---

```
Procedure:    VolumeMetamorphosis-FieldMorphing;  
Input:      Va, Vb: Volume; Ca, Cb: Control; L: Integer;  
Output:     Vout[1], Vout[2], ..., Vout[L]: Volume;
```

---

```
for k:=1 to L do  
    t = k / (L + 1.0);  
    Ck := InterpolateControlData(Ca, Cb, t);  
    Va := DistortVolume(Va, Ca, Ck);  
    Vb := DistortVolume(Vb, Cb, Ck);  
    Vout[k] := InterpolateVolume(Va, Vb, t);  
end for;
```

---

The key features of each volume are defined using a set of control fields. For the  $k^{th}$  inbetween volume, a set of inbetween control fields  $\mathbf{C}_k$  is first obtained, and is then used to distort  $\mathbf{V}_a$  and  $\mathbf{V}_b$ .

The volume distortion operates as an inverse mapping [3] from a *destination* volume  $\mathbf{V}_d$  back to a *source* volume  $\mathbf{V}_s$  that may be  $\mathbf{V}_a$  or  $\mathbf{V}_b$  in this case. Given

two corresponding sets of control fields, each pair of fields,  $\mathbf{c}_{d,j} \in \mathbf{C}_d$  and  $\mathbf{c}_{s,j} \in \mathbf{C}_{s,j}$ ,  $[j = 1, 2, \dots, m]$ , determines a coordinate mapping

$$\mathbf{p}_{i,j} \leftarrow \mathbf{f}_j(\mathbf{x}_{d,i}, \mathbf{c}_{d,j}, \mathbf{c}_{s,j}), [i = 1, 2, \dots, n], [j = 1, 2, \dots, m] \quad (4)$$

from each voxel  $(\mathbf{x}_{d,i}, \mathbf{v}_{d,i}) \in \mathbf{V}_d$  to a point  $\mathbf{p}_{i,j}$  relative to  $\mathbf{V}_s$ . Associated with each coordinate mapping is a weight  $\omega_j(\mathbf{x}_{i,j})$  that is normally computed according to the distance between  $\mathbf{x}_{i,j}$  and  $\mathbf{c}_{d,j}$ . For each voxel  $(\mathbf{x}_{d,i}, \mathbf{v}_{d,i})$  a point  $\mathbf{p}_i$  is obtained as

$$\mathbf{p}_i \leftarrow \sum_{j=1}^m \omega_j(\mathbf{x}_{i,j}) \cdot \mathbf{p}_{i,j}, \text{ with} \quad (5)$$

$$\sum_{j=1}^m \omega_j(\mathbf{x}_{i,j}) = 1, [i = 1, 2, \dots, n],$$

and a voxel in  $\mathbf{V}_s$  nearest  $\mathbf{p}_i$  is then identified and its values are assigned to  $\mathbf{v}_{d,i}$ . When volumes are represented by 3D regular grids, it is usually more accurate to identify the corner voxels of the cube containing  $\mathbf{p}_i$  and tri-linearly interpolate their values.

### 3.2.1 Point Fields

Two types of fields, namely *point* [8] and *line* [4], are often employed in 2D image distortion. A volume morphing method can be easily derived from the image morphing method using point fields. Let each control field be simply the coordinates of a 3D point. A simple mapping function

$$\mathbf{p}_{i,j} \leftarrow \mathbf{f}_j(\mathbf{x}_{d,i}, \mathbf{c}_{d,j}, \mathbf{c}_{s,j}) = \mathbf{c}_{s,j} + \mathbf{x}_{d,i} - \mathbf{c}_{d,j} \quad (6)$$

defines a geometric translation from  $\mathbf{x}_{i,j}$  to  $\mathbf{p}_{i,j}$ . A weighting function, with a slight modification to the one proposed in [8], can be written as

$$\omega_j(\mathbf{x}_{i,j}) = \frac{\sigma(\mathbf{x}_{i,j}, \mathbf{c}_j)}{\sum_{k=1}^m \sigma(\mathbf{x}_{i,k}, \mathbf{c}_k)}, \text{ with} \quad (7)$$

$$\sigma(\mathbf{x}_{i,j}, \mathbf{c}_j) = \left[ \frac{1}{\mathbf{a} + \mathbf{b} \cdot \|\mathbf{x}_{i,j} - \mathbf{c}_j\|} \right]^d,$$

where  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{d}$  are three user-defined constants whose values are in the ranges of  $(0, \infty)$ ,  $[0, \infty)$ , and  $[0, \infty)$  respectively. The main use of the constant  $\mathbf{a}$  is to keep the denominator from becoming zero. When  $\mathbf{b}$  is equal to or barely greater than zero, the weight of a point field at different voxels is hardly differentiated by the distances from the field to these voxels. The greater the value of  $\mathbf{b}$ , the more differentiation between different distances. The constant  $\mathbf{d}$  is used to control the relative influence which each point field has upon a voxel. When  $\mathbf{d}$  is of a large value, the voxel is likely to be affected only by the strong and nearby fields.

Inevitably this method inherits from its 2D progenitor the inability to control any transformation other than simple translation. The results produced by simple

point field morphing are generally no better than those by cross-dissolving methods. In order to perform more complex distortion, Ruprecht et al [9] introduced a 3×3 transformation matrix  $\mathbf{T}_j$  into (6), resulting in

$$p_{i,j} \leftarrow f_j(\mathbf{x}_{d,i}, \mathbf{c}_{d,j}, \mathbf{c}_{s,j}) = \mathbf{c}_{s,j} + \mathbf{T}_j \cdot (\mathbf{x}_{d,i} - \mathbf{c}_{d,j}). \quad (8)$$

The elements of  $\mathbf{T}_j$  are determined by solving a set of error functions. An alternative method which was also described in [9] is to construct the interpolant as a linear combination of basis functions whose coefficients are determined using control points.

### 3.2.2 Line Fields

The extension of 2D line fields to three dimensions is slightly less straightforward, as a coordinate mapping from a voxel in  $\mathbf{V}_d$  to one in  $\mathbf{V}_s$  cannot be consistently defined by a pair of line segments. The problem can be solved by using a supplementary vector in each line field.

A 3D line field consists of two end-points  $\mathbf{c}_1$  and  $\mathbf{c}_2$  of a line segment, and a supplementary vector  $N$ . With the field, a Euclidean coordinate system can be established, which is centred at  $\frac{\mathbf{c}_1 + \mathbf{c}_2}{2}$  with an orthonormal basis ( $U$ ,  $V$ ,  $W$ ) defined as

$$U = \frac{N}{\|N\|}, \quad V = \frac{N \times (\mathbf{c}_2 - \mathbf{c}_1)}{\|N \times (\mathbf{c}_2 - \mathbf{c}_1)\|}, \quad W = U \times V. \quad (9)$$

Given a pair of line fields,  $\mathbf{L}_{s,j}$  and  $\mathbf{L}_{d,j}$ , a voxel located at  $\mathbf{x}_{d,i}$  in  $\mathbf{V}_d$  can be mapped onto a point  $p_{i,j}$  relative to  $\mathbf{V}_s$  as follows

$$\begin{aligned} dp_{i,j} &\leftarrow f_{coord}(\mathbf{x}, \mathcal{E}(\mathbf{V}_d), \mathcal{E}(\mathbf{L}_{d,j})), \\ sp_{i,j} &\leftarrow dp_{i,j} \frac{\|\mathbf{c}_{s,2} - \mathbf{c}_{s,1}\|}{\|\mathbf{c}_{d,2} - \mathbf{c}_{d,1}\|}, \quad [i = 1, 2, \dots, n], [j = 1, 2, \dots, m], \quad (10) \\ p_{i,j} &\leftarrow f_{coord}(sp_{i,j}, \mathcal{E}(\mathbf{L}_{s,j}), \mathcal{E}(\mathbf{V}_s)), \end{aligned}$$

where  $\mathcal{E}(\mathbf{X})$  represents the Euclidean coordinate system associated to  $\mathbf{X}$  that may be either a volume or a line field, and the function  $f_{coord}(\mathbf{x}, \mathcal{E}_a, \mathcal{E}_b)$  transforms a point from one coordinate system to another. The weight of each mapping may be calculated using the following weighting functions

$$\begin{aligned} \omega_j(\mathbf{x}_{i,j}) &= \frac{\sigma(\mathbf{x}_{i,j}, \mathbf{L}_j)}{\sum_{k=1}^m \sigma(\mathbf{x}_{i,k}, \mathbf{L}_k)}, \quad \text{with} \\ \sigma(\mathbf{x}_{i,j}, \mathbf{L}_j) &= \left[ \frac{\|\mathbf{c}_{j,2} - \mathbf{c}_{j,1}\|^c}{\mathbf{a} + \mathbf{b} \cdot \delta(\mathbf{x}_{i,j}, \mathbf{c}_{j,1}, \mathbf{c}_{j,2})} \right]^d, \quad (11) \end{aligned}$$

where used-defined constants  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{d}$  have the same meaning as with a point field. The constant  $\mathbf{c}$ , usually in the range of  $[0, 1]$ , is used to moderate the strength of line fields which is proportional to the length of the line segments. When  $\mathbf{c}$  is zero, all fields have the same strength regardless of their length. Let  $\mathbf{u} = \frac{(\mathbf{x} - \mathbf{c}_1) \bullet (\mathbf{c}_2 - \mathbf{c}_1)}{\|\mathbf{c}_2 - \mathbf{c}_1\|^2}$ . The function  $\delta(\mathbf{x}, \mathbf{c}_1, \mathbf{c}_2)$  computes the distance from a point to a line segment as follows

$$\delta(\mathbf{x}, \mathbf{c}_1, \mathbf{c}_2) = \begin{cases} \|\mathbf{x} - \mathbf{c}_2\| \frac{|(\mathbf{x} - \mathbf{c}_1) \bullet (\mathbf{c}_2 - \mathbf{c}_1)|}{\|\mathbf{x} - \mathbf{c}_1\| \cdot \|\mathbf{c}_2 - \mathbf{c}_1\|} & \text{if } 0 < \mathbf{u} < 1; \\ \|\mathbf{x} - \mathbf{c}_1\| & \text{if } \mathbf{u} \leq 0; \\ \|\mathbf{x} - \mathbf{c}_2\| & \text{if } \mathbf{u} \geq 1. \end{cases} \quad (12)$$

The conditions where a line segment is degenerated to a point or it is parallel to  $N$  may be dealt with by introducing a second supplementary vector or simply disallowing such a condition.

### 3.2.3 Disk Fields

Alternatively, a *disk* field can be established with a centre point  $\mathbf{c}$ , a normal vector  $N$  and a radial vector  $\mathbf{R}$ , and it gives more control over three dimensional transformations in comparison with a line field. Any arbitrary point may thereby be represented in cylindrical coordinates  $[\zeta, \rho, \varphi]$  with respect to the disk field  $\mathbf{D}=[\mathbf{c}, N, \mathbf{R}]$ . Each pair of corresponding disk fields,  $\mathbf{D}_{s,j}$  and  $\mathbf{D}_{d,j}$  defines a coordinate mapping that can be used to control the distortion from  $\mathbf{V}_s$  to  $\mathbf{V}_d$  as

$$\begin{aligned} [\zeta_{i,j}, \rho_{i,j}, \varphi_{i,j}] &\leftarrow f_{eu\_to\_cy}(\mathbf{x}_{d,i}, \mathcal{E}(\mathbf{V}_d), \mathcal{E}(\mathbf{D}_{d,j})), \\ \zeta'_{i,j} &\leftarrow \zeta_{i,j} \frac{\|N_s\|}{\|N_d\|}, \quad \rho'_{i,j} \leftarrow \rho_{i,j} \frac{\|R_s\|}{\|R_d\|}, \quad \varphi'_{i,j} \leftarrow \varphi_{i,j}, \\ \mathbf{p}_{i,j} &\leftarrow f_{cy\_to\_eu}([\zeta'_{i,j}, \rho'_{i,j}, \varphi'_{i,j}], \mathcal{E}(\mathbf{D}_{s,j}), \mathcal{E}(\mathbf{V}_s)), \\ &[i = 1, 2, \dots, n], [j = 1, 2, \dots, m]. \end{aligned} \quad (12)$$

In (12),  $\mathcal{E}(\mathbf{D})$  gives a cylindrical coordinate system defined by a disk field, and  $f_{eu\_to\_cy}$  and  $f_{cy\_to\_eu}$  facilitate the transformations of a point between a Euclidean system and a cylindrical system. A weighting function similar to (11) may be used for disk field. A detail description of this method can be found in [10].

Other types of fields may also be constructed by varying the primary shapes and their influence upon voxels. Using a combination of different fields is also feasible, and some interesting work has been reported recently [11]. The results of morphing with line and disk fields are generally of satisfactory quality, but the extra degree of freedom introduces difficulties into the process of specifying a volume morphing. Without a good user interface, defining a reasonable number of fields is likely to require some arduous work.

### 3.3 Mesh Warping

Mesh warping [3] is the most popular method used for image metamorphosis. The control dataset associated with each image specifies a planar subdivision over the image, typically a parametric grid or a triangular mesh. Extrapolating from the method to a three dimensional analogue, a volume warping method can be derived. Volumes  $\mathbf{V}_a$  and  $\mathbf{V}_b$  are first partitioned respectively by two spatial subdivisions that are of an equal number of elements. Deformed subdivisions are then obtained for inbetween volumes by interpolation. A voxel in an inbetween volume is mapped onto a voxel in each of  $\mathbf{V}_a$  and  $\mathbf{V}_b$ . Similar to field morphing, the values of voxels in the inbetween volume are determined by linearly interpolating those of the corresponding voxels in  $\mathbf{V}_a$  and  $\mathbf{V}_b$ .

It is the distortion algorithms employed by mesh warping distinguish it from field morphing. The elements of a subdivision, mostly the vertices, are used to define coordinate mappings. In addition to this, some elements, mostly the polyhedra, are also used to restrict the control of each coordinate mapping to a very small set of elements. A variety of image warping algorithms [3] have been developed and most are often designed specifically for particular type of meshes. The 3D extensions of two widely used image warping algorithms [12-14] are described below.

#### 3.3.1 Parametric Grid

Consider a given source volume  $\mathbf{V}_s$ , a destination volume  $\mathbf{V}_d$  to be constructed, and two subdivisions  $\mathbf{G}_s$  and  $\mathbf{G}_d$  which are defined by two 3D parametric grids. Each grid consists of  $m$  vertices, and partitions a volume into a contiguous set of eight-cornered bricks. The bricks may be fitted with either linear or curved edges and surfaces. With such grids, a 3-pass warping algorithm can be used to map all voxels in  $\mathbf{V}_d$  to those in  $\mathbf{V}_s$  with two intermediate volumes  $\mathbf{V}_{i1}$  and  $\mathbf{V}_{i2}$ . The 3 passes of the algorithms are

- first pass — where an intermediate grid  $\mathbf{G}_{i1}$  is constructed such that

$$\mathbf{G}_{i1} = \left\{ v_j = \begin{bmatrix} x_{d,j} & y_{s,j} & z_{s,j} \\ x_{d,j} & y_{d,j} & z_{d,j} \end{bmatrix} \left| \begin{array}{l} [x_{s,j}, y_{s,j}, z_{s,j}] \in \mathbf{G}_s, \\ [x_{d,j}, y_{d,j}, z_{d,j}] \in \mathbf{G}_d, \\ [j = 1, 2, \dots, m] \end{array} \right. \right\}. \quad (13)$$

With respect to  $\mathbf{G}_s$ , each of the eight-cornered bricks in  $\mathbf{G}_{i1}$  can be deformed only in the x direction. The intermediate volume  $\mathbf{V}_{i1}$  is then obtained by interpolating its voxels along scan-lines parallel to the x-axis using  $\mathbf{G}_s$  and  $\mathbf{G}_{i1}$ .

- second pass — where  $\mathbf{G}_{i2}$  is obtained with  $v_j = [x_{d,j}, y_{d,j}, z_{s,j}]$  and  $\mathbf{V}_{i2}$  is constructed from  $\mathbf{V}_{i1}$  along scan-lines parallel to the y-axis under the control of  $\mathbf{G}_{i1}$  and  $\mathbf{G}_{i2}$ .

- third pass — where  $\mathbf{V}_d$  is constructed from  $\mathbf{V}_{i2}$  along scan-lines parallel to the z-axis under the control of  $\mathbf{G}_{i2}$  and  $\mathbf{G}_d$ .

### 3.3.2 Tetrahedral Mesh

Given a set of points, a tetrahedral mesh may be constructed using a 3D triangulation algorithm. To construct a destination volume  $\mathbf{V}_d$  from a given source volume  $\mathbf{V}_s$  and two corresponding meshes  $\mathbf{T}_s$  and  $\mathbf{T}_d$ , each voxel in  $\mathbf{V}_d$  is first mapped to a set of bary-centric coordinates  $[\beta^1, \beta^2, \beta^3, \beta^4]$  with respect to the tetrahedron  $\mathbf{t}_{d,j} \in \mathbf{T}_d, [j = 1, 2, \dots, m]$  containing the voxel as follows

$$\beta_{i,j}^k \leftarrow \frac{\Delta(\dots, \mathbf{t}_{d,j}^{k-1}, \mathbf{x}_{d,i}, \mathbf{t}_{d,j}^{k+1}, \dots)}{\Delta(\mathbf{t}_{d,j}^1, \mathbf{t}_{d,j}^2, \mathbf{t}_{d,j}^3, \mathbf{t}_{d,j}^4)}, [k = 1, 2, 3, 4], \text{ with}$$

$$\Delta(v^1, v^2, v^3, v^4) = \det \begin{bmatrix} v_x^1 & v_y^1 & v_z^1 & 1 \\ v_x^2 & v_y^2 & v_z^2 & 1 \\ v_x^3 & v_y^3 & v_z^3 & 1 \\ v_x^4 & v_y^4 & v_z^4 & 1 \end{bmatrix}, \quad (14)$$

where  $\mathbf{t}_{d,j}^k$  is the  $k^{\text{th}}$  vertex of  $\mathbf{t}_{d,j}$ . The bary-centric coordinates are then transformed back to Euclidean coordinates as

$$\mathbf{p}_{i,j} \leftarrow \sum_{k=1}^4 \beta_{i,j}^k \cdot \mathbf{t}_{s,j}^k. \quad (15)$$

With both methods, it is however necessary to make sure that there is no element whose corresponding element has a "fold-over" structure. With the warping approach, distortion is constrained by individual element, and it is therefore relatively easier to achieve desired transformation without causing "ghost shadows". In most cases, they require much more control elements than field morphing methods.

## 4 Comparison

For each of the methods discussed above, Table 1 summarises the worst-case time complexity, allowed operations, usability and the general quality of its results,

Table 1. Comparison of volume morphing methods.

Method	Complexity	Operations	Results	Usage
Linear Interpolation	$O(n)$	C	poor	very easy
Fourier Interpolation	$O(n \cdot \log n)$	C	mostly poor	fairly easy
Simple Point Field	$O(n \cdot m)$	C,D,T	poor	easy
Point Field with T	$O(n \cdot m + m^2)$	C,D,T,S,R	average	average
Vector Field	$O(n \cdot m)$	C,D,T,S,R	average	difficult
Disk Field	$O(n \cdot m)$	C,D,T,S,R	good	difficult
Grid Warping	$O(n+m)$	C,D,T,S,R	good	difficult
Tetra-Mesh Warping	$O(n+m)$	C,D,T,S,R	average	average

In the above we assume the volume size is  $n$ , and the size of control datasets is  $m$ . Operations considered are cross-dissolving (C), one-way distortion (D), translation (T), scaling (S) and rotation (R).

A subset of these algorithms, including cross-dissolving with linear interpolation, simple point field morphing, disk field morphing and tetrahedral mesh warping, have been implemented in C on a DEC Alpha workstation. Results were rendered using the Advanced Visualisation System (AVS).

One of the tests is the metamorphosis from a volume containing a cube as shown in Figure 1(a) and one containing a sphere in Figure 1(b). Both volumes represent continuous functions from high voxel values at the volume centre to low values around the boundary. Four sets of inbetween volumes generated with different methods are given in Figure 2, 3, 4 and 5 respectively. Both cross dissolving and simple point field morphing produced reasonable results in this case. However, the results would have been of a much poorer quality had the given volumes contained discrete solid objects. With tetrahedral mesh warping, some undesirable visual artifacts are obvious as the mesh does not seem fine enough though 96 tetrahedra were used to produce this sequence. The most satisfactory results were produced using disk field morphing.

Another test carried out is to transform a sphere volume to a volume containing a CT (Computed Tomography) scan of a head, as shown in Figure 6. Because the head was not placed at the centre of the volume, the metamorphosis involves translation as well as deformation of the volumes. As shown in Figure 7, the cross-dissolving method was incapable of accomplishing such a task. On the other hand, with 12 control fields, disk morphing produced some satisfactory results.

Some of the execution times obtained during testing are listed in Table 2, where the number of fields or tetrahedra is fixed at 10 for all methods except for cross-dissolving with linear interpolation.

Table 2. the run-times (1/60 sec.) of volume morphing algorithms.

Method	16 <sup>3</sup>	32 <sup>3</sup>	64 <sup>3</sup>	128 <sup>3</sup>
Linear Interpolation	<1	<1	12	106
Simple Point Field	43	343	2782	22184
Disk Field	140	1120	8979	71953
Tetra-Mesh Warping	20	146	1125	8860

## 5. Conclusions

Several volume morphing methods have been described in this paper. The tests of these methods has shown that some of them are able to produce with good quality morphing results, even with relatively simple control datasets. Like image-based metamorphosis, the volume morphing technique can be found useful in a variety of applications, especially in the synthesis of 3D models for creating images sequences in video production. Future work in this area undoubtedly includes the development of methods that are fast to run, easy to use, and able to produce smooth and consistent morphing results. There is also a need to establish a set of benchmark problems and define a set of quantifiable attributes in order to carry out a more accurate comparison. With volume data, it is beneficial, sometimes necessary, to use a volume morphing method rather than morphing individual slices of the volumes or interpolates the isosurfaces of the volumes. Therefore, the most important task perhaps is to make sophisticated tools for volume morphing available to users as soon as possible.

## References

1. Kaufman A, Bakalash R. Memory and processing architecture for 3D voxel-based imagery. *IEEE Computer Graphics and Applications* 1988; 8(6):10-23
2. Meagher D. Applying solids processing to medical planning. In: *Proc. NCGA'85*, Dallas, 1985, pp 101-109
3. Woldberg G. *Digital image warping*. IEEE Computer Society Press, 1990
4. Beier T, Neely S. Feature-based image metamorphosis. *SIGGRAPH Computer Graphics* 1992; 26(2):35-42
5. Elvins TT. A survey of algorithms for volume visualisation. *SIGGRAPH Computer Graphics* 1992; 26(3):194-200
6. Herman GT, Liu HK. Three dimensional display of human organs from computed tomograms. *Computer Graphics and Image Processing* 1979; 9(1):1-21
7. Hughes JF. Scheduled Fourier volume morphing. *SIGGRAPH Computer Graphics* 1992; 26(2):43-45
8. Shepard, D. A two-dimensional interpolation function for irregularly spaced data. In: *Proc. 23rd National Conference of ACM*, 1968, pp 517-524
9. Ruprecht D, Nagel R, Müller H. Spatial free form deformation with scattered data interpolation methods. *Research Report: 539*, Department of Computer Science, University of Dortmund, Germany, 1994

10. Chen M, Jones MW, Townsend P. Volume morphing using disk field. Research Report: CSR 28-94, Department of Computer Science, University of Wales Swansea, UK, 1994
11. Leros A. 3D volume morphing. World Wide Web page, Stanford University, USA (<http://www-graphics.stanford.edu/~tolis/morph.html>), 1994
12. Smythe DB. A two-pass mesh warping algorithm for object transformation and image interpolation. ILM technical Memo: #1030, Computer Graphics Department, Lucasfilm Ltd., 1990
13. Goshtasby A. Piecewise linear mapping functions for image registration. Pattern Recognition 1986; 19(6):459-466
14. Goshtasby A. Piecewise cubic mapping functions for image registration. Pattern Recognition 1987; 20(5):525-533

Figure 1. (a) A volume representing a cube function, and  
(b) a volume representing a sphere function.

Figure 2. Three inbetween volumes generated using the cross-dissolving method with linear interpolation.

Figure 3. Three inbetween volumes generated using the simple point field morphing method.

Figure 4. Three inbetween volumes generated using the disk field morphing method.

Figure 5. Three inbetween volumes generated using the tetrahedral mesh warping method.

Figure 6. (a) A volume representing a sphere function, and (b) a CT scan of head.

Figure 7. Cross-dissolving from the sphere to the head.

Figure 8. Disk field morphing from the sphere to the head.