http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.4, No.6, P.637-642, Nov.-Dec., 2003

Efficient volume preserving approach for skeleton-based implicit surfaces*

SHI Hong-bing(史红兵), TONG Ruo-feng(童若锋)†, DONG Jin-xiang(董金祥)

(*State Key Laboratory of CAD & CG, Institute of Artificial Intelligence, Department of Computer Science and Engineering, Zhejiang University, Hangzhou 310027, China*)

†E-mail: trf@cs.zju.edu.cn

Received Dec.2, 2002; revision accepted June 4, 2003

Abstract: This paper presents an efficient way to preserve the volume of implicit surfaces generated by skeletons. Recursive subdivision is used to efficiently calculate the volume. The criterion for subdivision is obtained by using the property of density functions and treating different types of skeletons respectively to get accurate minimum and maximum distances from a cube to a skeleton. Compared with the criterion generated by other ways such as using traditional Interval Analysis, Affine Arithmetic, or Lipschitz condition, our approach is much better both in speed and accuracy.

Key words: Volume preserving, Skeleton-based implicit surface, Subdivision criterion, Interval analysis
Document code: A **CLC number:** TP391.72

INTRODUCTION

Implicit surfaces, especially skeleton-based ones, are widely used in modeling soft objects such as organs, clouds and liquids because of their capability for generating smooth surfaces of arbitrary geometry and topology (Fujita *et al.*, 1990; Yu *et al.*, 1999; Murta and Miller, 1999). They are also effective in controlling the deformation and movement of objects, because when skeletons move, the generated surface follows them automatically. Thus, the deformation of objects can be easily described by the movement of skeletons (Desbrun and Gasuel, 1995; O'Brien and Hodgins, 1995). But a problem when using implicit surfaces to simulate animating objects is how to preserve the volume while the objects undergo deformation.

All objects made of imcompressible substances will preserve their volume when they deforming. This drives many researchers to work on the volume-preserving problem. Sederberg and Parry (1986) pointed out that in Free-Form Deformation, all objects inside the frame will preserve their volume when the Jacobian of the deformation function is equal to 1. The problem is, however, that ensuring the Jacobian to be 1 while the users move control points is difficult. In Rappoport *et al.* (1996), volume preservation is solved for objects defined by Bezier solids. Aubert and Bechmann (1997) presented a volume-preserving deformation method suitable for polyhedral objects. And Hirota *et al.* (2000) presented an efficient algorithm to preserve the total volume of a solid undergoing free-form deformation using a multi-level optimization. But there are few approaches available for implicit surfaces. The only approach we have found is that presented in (Murta and Miller, 1999) where the maximum density is used as the parameter to adjust the volume. However, their method is inefficient for calculating volume.

Several parameters of implicit surface can be used to adjust the volume. But no matter using which parameter, volume calculation will be used from time to time in the whole procedure. Thus the efficiency of volume calculation is very important. The volume of an object defined by $f(x, y, z) > 0$ can be described as $\iiint_{f(x,y,z)>0} \mathrm{d}x\mathrm{d}y\mathrm{d}z$. This integral expression of volume cannot be computed analytically for most field functions, which

* Project supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, and the Natural Science Foundation (No.601107) of Zhejiang Province, China

makes us turn to numerical ways. The simplest of which is to calculate the volume numerically by uniform sampling, but recursive subdivision is usually more efficient. And the efficiency of the recursive subdivision approach is determined by the accuracy and computation cost of the criterion of subdivision. Interval Analysis(Snyder, 1992), Affine Arithmetic(Comba and Stolfi, 1993), and Lipschitz condition(Galin and Akkouche, 2000) can be used to generate subdivision criterion, but their efficiency still cannot satisfy the requirement for volume preserving. The main contribution of this paper is presenting an efficient and accurate approach to generate the subdivision criterion.

MAIN PROCEDURE FOR VOLUME-PRESERVING

The surface of an object generated by skeletons is defined by the points satisfying the following equation:

$$f(x,y,z) = \sum_{i=0}^{n} q_i f_i - T_0 = 0 \quad (1)$$

Where T_0 is a threshold, q_i is a factor coefficient (also called the maximum density) of Skeleton I, f_i is the density function of Skeleton I. A typical density function was proposed by Murakami as Eq.(2).

$$f_i(r) = \left(1 - \left(\frac{r}{R_i}\right)^2\right)^2 \quad (2)$$

Where r is the distance from point (x, y, z) to the skeleton, R_i is the influence radius of skeleton I.

Actually there are many existing forms of density functions that can be used in Eq.(1). They have the common properties that $f_i(0) = 1$, $f_i(1) = 0$, $f_i'(0) = f_i'(1) = 0$, as illustrated in Fig.1. These properties make the density generated by several skeletons C^1 continuous.

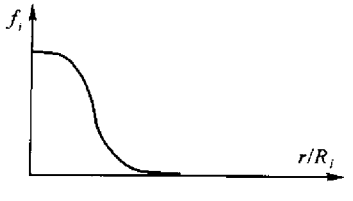

Fig.1 Shape of field function

In order to make the approach suitable for all implicit surfaces generated by any form of density function, we discard the above detailed equations and use their common properties illustrated by Fig.1.

From Eq.(1), we know that changing q_i, f_i or T_0 will change the region where $f(x, y, z) > 0$ and then change the volume. f_i is defined by r and R_i, with r being the distance from point (x, y, z) to the skeleton I, so the position of skeleton I is also essential to the volume. Usually the position and direction of skeletons is used to control the deformation and movement of the objects; changing them will distort the animation effects; so that parameters we can use to adjust the volume are q_i, R_i and T_0. T_0 is monotonic to f, allowing for simple control of volume. The problem is that when T_0 changes, the entire implicit surface will also change, which fails in local volume controlling. So we have to fix T_0 and search for other volume control parameters.

Murta and Miller (1999) used the maximum density q_i as a volume-controlling parameter. In this way, the volume can be increased by increasing q_i. There is still a drawback: when q_i has been assigned a large value, it exerts little influence on the volume. And no matter how large q_i is, the iso-surface cannot expand beyond its influence range. Therefore, sometimes we cannot adjust the current volume to its original value with the skeletons' influence radii unchanged. For example, when ten separated metaballs (implicit surfaces generated by point skeletons) arrive at the same location and generate one isolated surface, its volume cannot be adjusted to the original volume no matter how large the values q_is are assigned, as the iso-surface cannot expand beyond the ball with the influence radius.

To overcome these disadvantages, we take both q_i and R_i as control parameters simultaneously to adjust the volume. Both of them are monotonic to the volume, so when we want to increase the volume, we simply increase the maximum density q_i and the influence radius R_i at the same time.

The main procedure of volume-preserving animation for a skeleton-based implicit surface is

as follows:

Step 1. Initialization: such as setting scene and time interval, generating skeletons, and calculating the initial volume V_0 enclosed by the surfaces.

Then, to each time step t_i, perform the following steps:

Step 2. Move skeletons to new positions according to the demands of animation effects.

Step 3. Calculate the volume V_i of the object according to the new positions of skeletons, then adjust the volume-controlling parameter q_k and R_k of every skeleton recursively to make the current volume V_i equal to V_0. R_k and q_k are adjusted by $R_k = (1 + \mu) * R_k$ and $q_k = (1 + \mu) * q_k$, where μ is a parameter to adjust all the R_ks and q_ks. A binary search is used to find the suitable value of μ.

Step 4. Rendering.

In the above procedure, Step 1, Step 2, and Step 4 are common for a skeleton-based animation, while Step 3 is special for volume-preserving; its flow is illustrated in Fig.2.

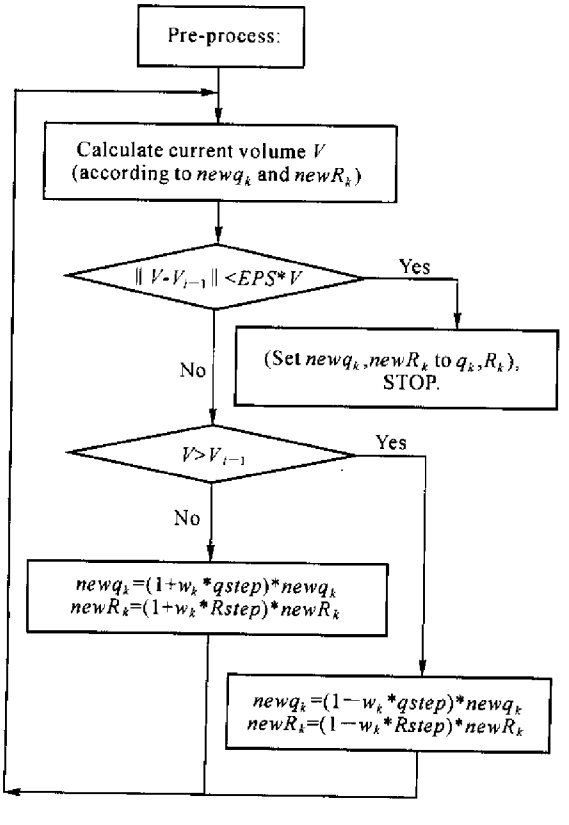

Fig.2 Flow of volume preserving

In Fig.2, the preprocess module executes all the preparation jobs, such as:

Calculate the previous volume V_{i-1}.

Set a weight w_k to each skeleton K. (w_k is set to control the skeleton's contribution to volume variation, when controlling the volume, the bigger w_k is, the more the volume controlling parameters q_k and R_k will change). Set the original volume controlling parameters.

Special attention must be paid to the fact that *qstep* and *Rstep* will become smaller when current volume V is approaching to previous volume V_{i-1}. Firstly, *Rstep* and *qstep* are set with *coef* $\frac{V - V_{i-1}}{V}$. Use these *Rstep* and *qstep* to calculate the new q_i and adjust the volume repeatedly until $V - V_{i-1}$ changes its sign. Then *Rstep* and *qstep* are set by half of themselves. Repeat this procedure until $\left|\frac{V - V_{i-1}}{V}\right| > EPS$. This midpoint approach makes V converge to V_{i-1} stably, although at a low efficiency. But we still profit a lot from this compared with the time-consuming constrained minimization. The main reason why we can use such a simple flow to preserve the volume is that there are only two parameters to control the volume and both are monotonic to the volume.

SUBDIVISION CRITERION FOR VOLUME CALULATION

The volume enclosed by an implicit surface cannot be calculated analytically. Recursive subdivision is employed as a numerical way to calculate the volume because it is more efficient than uniform sampling. Its efficiency in practical use is determined by the accuracy and speed of judging whether a cube X is inside or outside the object. An inaccurate judgment will cause unnecessary subdivision and lowers the efficiency. Interval Analysis is a valid mathematic tool to help solve the inside-outside problem more accurately.

Interval Analysis

Interval Analysis (Hirota *et al.*, 2000) (also called Interval Arithmetic, IA) was originally proposed by Moore in 1966. It can control approximate errors by truncation and round off, and process the behavior of functions efficiently and reliably over whole sets of arguments at a

time. Recently, this advantage has drawn the attention of many computer graphics researchers and been put to use in surface intersection, ray tracing, 2D implicit curve contouring and other CG problems.

An Interval, $A = [a, b]$, is a subset of R defined as $[a, b] = \{x \mid a \leqslant x \leqslant b, x, a, b \in R\}$, with a and b called the bounds of the interval; a is the lower bound and b is the upper bound. In the following sections, capital letters are used to represent an interval, while the upper and lower bounds of the interval X are denoted respectively as $X.ub$ and $X.lb$.

Interval Arithmetic generalizes an ordinary arithmetic to closed, bounded ranges of real numbers. All the variables of the function are intervals, and correspondingly, the function is also an interval, and called an inclusion function. Let $\{f(x) \mid (x \in X)\}$ be a traditional function, and the inclusion function for f, written as F, can be described as $F(X) = [F.lb, F.ub]$, where $F.lb \leqslant \min_{x \in X} f(x)$, $F.ub \geqslant \max_{x \in X} f(x)$ Many possible inclusion functions can be defined for a given function f. Obviously, an ideal function is that, $F.lb = \min_{x \in X} f(x)$, $F.ub = \max_{x \in X} f(x)$. But for a complicated function f, it is impractical to calculate the minimum and maximum of f. We must therefore compromise between the tightness of the bound and the calculation cost.

The natural interval extensions of the elementary operations of arithmetic are: $X + Y = [X.lb + Y.lb, X.ub + Y.ub]$, $X - Y = [X.lb - Y.ub, X.ub - Y.lb]$ and $X * Y = [\min(X.lb * Y.lb, X.lb * Y.ub, X.ub * Y.lb, X.ub * Y.ub), \max(X.lb * Y.lb, X.lb * Y.ub, X.ub * Y.lb, X.ub * Y.ub)]$. It is clear that these interval operations (called formulae IA) can be applied recursively to yield an inclusion function for an arbitrary, nested combination of arithmetic operations. The problem is that the range intervals produced in this way are often much wider than the true ranges of the function.

Inclusion function for density function generated by skeletons

In this section, we calculate the inclusion function $F(C)$ of density function $f(x, y, z) = \sum_{i=1}^{N} k_i f_i(x, y, z) = \sum_{i=1}^{N} k_i g_i(r) = \sum_{i=1}^{N} k_i \left(1 - \left(\frac{r}{R_i}\right)^2\right)^2$ on cell $C = [X, Y, Z]$, where $X = [X.lb, X.ub]$, $Y = [Y.lb, Y.ub]$, $Z = [Z.lb, Z.ub]$, $k_i > 0$. If the lower bound of $F(C)$, $F.lb$ is bigger than T_0, C is absolutely inside the implicit surface; if the upper bound of $F(C)$, $F.ub$ is smaller than T_0, C is absolutely out of the implicit surface.

Using formulae IA, we have $F(C) = [F.lb, F.ub] = \left[\sum_{i=1}^{N} k_i F_i.lb, \sum_{i=1}^{N} k_i F_i.ub\right]$, where $F_i(C) = [F_i.lb, F_i.ub]$ is the inclusion function of $f_i(x, y, z)$ on cell C. To avoid the expansion of range interval caused by using formulae IA for long computation chains, we calculate $F_i(C) = [F_i.lb, F_i.ub]$, and in particular, we utilize the property of density function to get much tighter range intervals. The ideal $F_i(C)$ is that $F_i.lb = \min_{x \in X, y \in Y, z \in Z} f_i(x, y, z)$ and $F_i.ub = \max_{x \in X, y \in Y, z \in Z} f_i(x, y, z)$. Note that $g_i(r)$ is a monotonic decrease function of r, thus the problem can be converted to finding the minimum and maximum of distance from points within C to the skeleton I. Because skeletons usually are composed of points and line segments, to calculate the accurate minimum and maximum distance, we treat the two types of skeletons respectively.

To a point skeleton, $\max_{x \in X, y \in Y, z \in Z} r^2 = \max_{x \in X, y \in Y, z \in Z} ((x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2)$ and $\min_{x \in X, y \in Y, z \in Z} r^2 = \min_{x \in X, y \in Y, z \in Z} ((x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2)$, where (x_i, y_i, z_i) is the position of skeleton I. Because x, y, z are independent variables, we have $\max_{x \in X, y \in Y, z \in Z} r^2 = \max_{x \in X}(x - x_i)^2 + \max_{y \in Y}(y - y_i)^2 + \max_{z \in Z}(z - z_i)^2$ and $\min_{x \in X, y \in Y, z \in Z} r^2 = \min_{x \in X}(x - x_i)^2 + \min_{y \in Y}(y - y_i)^2 + \min_{z \in Z}(z - z_i)^2$. Now the problem has become simplified, with $\max_{x \in X}(x - x_i)^2$ and $\min_{x \in X}(x - x_i)^2$ calculated by the following equations.

$$\max_{x \in X}(x - x_i)^2 = \begin{cases} (X.ub - x_i)^2 & \text{if } x_i < \dfrac{X.lb + X.ub}{2} \\ (X.lb - x_i)^2 & \text{otherwise} \end{cases} \tag{3}$$

$$\min_{x \in X}(x - x_i)^2 = \begin{cases} (X.ub - x_i)^2 & \text{if } x_i < X.lb \\ (X.ub - x_i)^2 & \text{if } x_i > X.ub \\ 0.0 & \text{otherwise} \end{cases} \tag{4}$$

Similarly, we can calculate $\max_{y\in Y}(y-y_i)^2$, $\max_{z\in Z}(z-z_i)^2$ and $\min_{y\in Y}(y-y_i)^2$, $\min_{z\in Z}(z-z_i)^2$, and then obtain $\max_{x\in X,y\in Y,z\in Z} r^2$ and $\min_{x\in X,y\in Y,z\in Z} r^2$.

To a line segment skeleton p_1p_2, $\max_{p\in C} r(p, p_1p_2) = \max(\max_{p\in C} r(p, p_1), \max_{p\in C} r(p, p_2))$. Thus the maximum distance to a line segment is converted to the distance to two points, and it is easy to be obtained by using the approach introduced in point type skeleton. To calculate the minimum distance from cube C to line segment p_1p_2 is a little difficult. First, we judge whether p_1p_2 intersects cube C. If so, $r(C, p_1p_2) = \min_{p\in C} r(p, p_1p_2) = 0.0$. Otherwise, we calculate the minimum distance by $r(C, p_1p_2) = \min(\min_{f\in F} r(f, p_1), \min_{f\in F} r(f, p_2), \min_{l\in L} r(l, p_1p_2))$, where F is the collection of six faces of the cube C, and L is the collection of eight edges of the cube C. In this way, we calculate the maximum and minimum distance from cube C to skeleton I, and obtain $F_i(C) = [F_i.lb, F_i.ub]$.

EXAMPLES

We have applied this volume preserving approach to simulate drops of water falling from a plane. The original scene of Fig.3 is composed of 58 metaballs on the same horizon plane. The maximum densities q_is of these metaballs are all assigned the value of 1.0. When drops fall from the plane, the part remaining on the plane shrinks to balance the entire volume with the original volume.

The example uses $f_s(r) = \left(1 - \frac{r^2}{R_s^2}\right)^2$ as the density function.

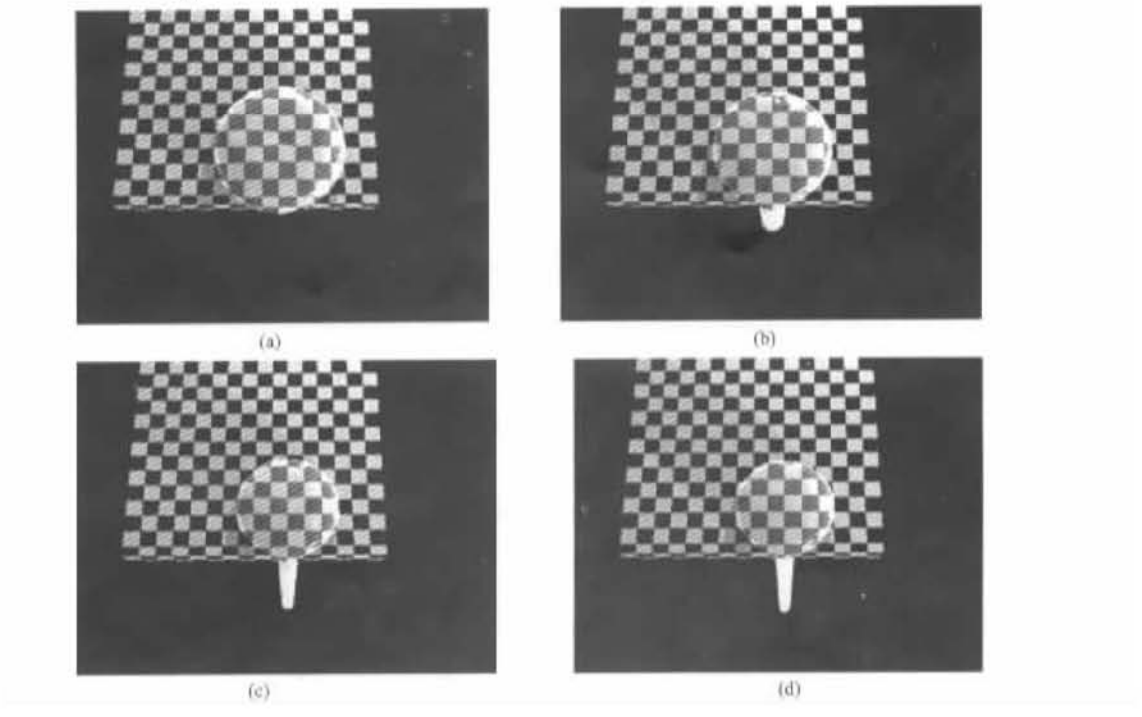

Fig.3 Volume-preserving animation

(a) Initial shape; (b) Intermediate shape 1; (c) Intermediate shape 2; (d) Final shape

CONCLUSIONS

In this paper, we have presented a volume-preserving approach for objects modeled by skeleton-based implicit surfaces. We adjust the volume of objects to an original volume by tuning the influence radius and maximum density of skeletons, thereby avoiding distortion of the animation effect, since in most animation, the deformation of objects is achieved by the movement of skeletons.

We use a recursive subdivision scheme to calculate the volume of implicit surfaces. A relatively accurate subdivision criterion is obtained by using the particular property of density function with small computation. Taking point skeletons as example, we can know from Section "Indusion function for density function generated by skeletons" that we obtained the range of density function within the entire cube only by the computation cost of calculating the function at two points. It is much better than using traditional Interval Analysis or Affine Arithmetic (Comba and Stolfi, 1993) both in speed and accuracy. It is also more advanced than the Lipschitz Condition approach(Galin and Akkouche, 2000).

The Lipschitz Condition approach is based on the following theorem:

To any continuous function f in interval C, $\exists \lambda > 0$, $\forall x_1, x_2 \in C$, $|f(x_1) - f(x_2)| \leqslant \lambda \| x_1 - x_2 \|$

λ is usually taken as the upper boundary of the derivative of $f(x)$ within $[x_1, x_2]$.

This approach expands to some extent the real range of f in C. As illustrated in Fig.4, to a single metaball, its range $|f(x_1) - f(x_2)|$ is expanded to $\lambda \| x_1 - x_2 \|$. This undoubtedly will lead to unnecessary subdivision and lower the efficiency; while our approach can rapidly calculate the accurate range of density function generated by a single metaball within a cube.

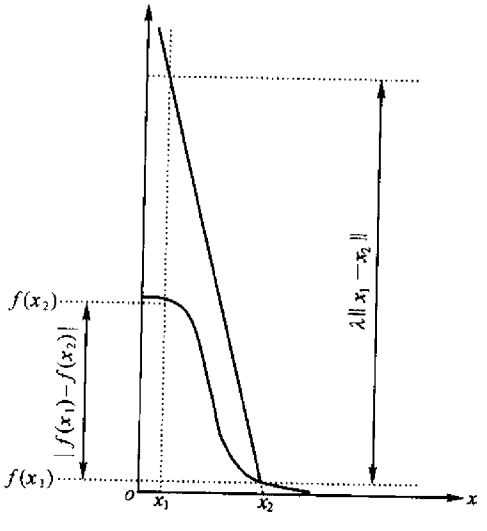

Fig.4 The difference between $\lambda \| x_1 - x_2 \|$ and $|f(x_1) - f(x_2)|$

Although our approach induces an expansion when calculating the range of muti-metaballs from the range of single metaball, the Lipschitz Condition approach(Foster and Metaxas, 1996) did the same thing in this stage. Thus, from the materials we obtained, our approach has advantages both in accuracy and speed comparing with the existing approaches.

We used Eq.(2) as a density function in our examples, but the approach can be applied to any field function having the shape of Fig.1.

References

Aubert, F. and Bechmann, D., 1997. Volume-Preserving Space Deformation. *Computer & Graphics*, **21**(5): 625 – 637.

Comba, J.L.D. and Stolfi, J., 1993. Affine Arithmetic and its Applications to Computer Graphics. Proceedings of the VI Sibgrapi, p.9 – 18.

Desbrun, M. and Gascuel, M.P., 1995. Animating Soft Substances with Implicit Surfaces. SIGGRAPH'95, p. 287 – 290.

Foster, N. and Metaxas, D., 1996. Realistic animation of liquids. *Graphical Models and Image Processing*, **58** (5): 471 – 483.

Fujita, T., Hirota, K. and Murakami, K., 1990. Representation of Splashing Water using metaball Model. *FUJITSU*, **41**(2): 159 – 165.

Galin, E. and Akkouche, S., 2000. Incremental Polygonization of Implicit Surfaces. *Graphical Models*, **62** (1): 19 – 39.

Hirota, G., Maheshwari, R. and Lin, M.C., 2000. Fast volume – preserving free-form deformation using multilevel optimization. *CAD*, **32**(8/9): 499 – 512.

Murta, A. and Miller, J., 1999. Modeling and Rendering Liquids in Motion. WSCG'99 Proceedings, p.194 – 201.

O'Brien, J.F. and Hodgins, J.K., 1995. Dynamic Simulation of Splashing Fluids. Computer Animation '95, p.198 – 205.

Rappoport, A., Sheffer, A. and Bercovier, M., 1996. Volume Preserving Free-Form Solids. *IEEE Transactions on visualization and computer graphics*, **2**(1): 19 – 27.

Sederberg, T.W. and Parry, S.R., 1986. Free-form deformation of solid geometric models. *Computer Graphics*, **20**(4): 151 – 160.

Snyder, J.M., 1992. Interval analysis for computer graphics. *Computer Graphics*, **26**(2): 121 – 130.

Yu, Y.J., Jung, H.Y. and Cho, H.G., 1999. A New Water Droplet Model Using Metaball in the Gravitational Field. Computer & Graphics, p.213 – 222.