The Visual Computer

A volume-preserving approach for modeling and animating water flows generated by metaballs

Ruofeng Tong[1],
Kazufumi Kaneda[2],
Hideo Yamashita[2]

[1] Department of Computer Science and Engineering, Zhejiang University, Hangzhou, 310027, China
[2] Intelligent Systems and Modeling Laboratory, Hiroshima University, Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8524, Japan
E-mail: kin@eml.hiroshima-u.ac.jp

Published online: 23 July 2002
© Springer-Verlag 2002

This paper presents a volume-preserving approach for animating liquid flows modeled by metaballs. A volume of liquid can be adjusted to a previous volume by using the influence radius and the maximum density of metaballs as volume-controlling parameters. Recursive subdivision is used to efficiently calculate the volume of implicit surfaces. The criterion for subdivision is obtained by using the concept of interval analysis and the common property of metaball density functions. Providing a sequence of parameters, the volume-compensation region can be controlled according to the substance making up the object, resulting in local preservation of the volume. Set partition is used for determining isolated surfaces in order to apply local-volume preservation.

Key words: Volume preserving – Metaballs – Recursive subdivision – Interval analysis – Graph theory

Correspondence to: K. Kaneda

1 Introduction

Fluid simulation has been an active field in computer graphics for some time. Besides the movement and appearance of large-scale liquids [9–11, 17, 19], small-scale liquids, such as water splashing and water droplets, have recently drawn the attention of many researchers [1, 8, 12, 13, 15, 18, 23]. Fujita et al. [13] presented a method for describing a drop of rain falling and producing a crown of splashing water; O'Brien et al. [18] developed a spray model using a particle system; to realistically simulate water droplets, Kaneda et al. [8] focused on their movement and rendering; Yu et al. [23] emphasized their shapes; and Fournier et al. [12] proposed approaches for both motion and shape behavior. Most of these examples use metaballs to model the small-scale liquids.

Since proposed by Blinn and Nishimura [3, 16], the metaball method has been widely used in modeling soft objects, such as clouds and liquids, because of its capability of generating smooth surfaces of arbitrary geometry and topology [13, 15, 23]. The method also is effective in modeling the deformation and movement of liquids. When metaballs move, the generated surface follows them automatically. Thus, the deformation of liquids is usually described by the movement of metaballs. Various models have been proposed to specify the motion scheme of the metaballs, from a purely keyframe-based system to physical animation. An appropriate motion-controlling approach can be selected according to the demands of animation. Moreover, it is quite natural to describe the merging and separating of liquids based on their blending property. But a problem still exists – that is, how to preserve volume while the liquids undergo deformation. This is precisely the issue we attempt to solve in this paper.

When the shape of liquids generated by metaballs changes from time step t_i (shape A) to time step t_{i+1} (shape B), we adjust shape B to make its volume equal to that of shape A. Shape B is adjusted under the constraint that the positions of all the metaballs generating it remain unchanged, because in most metaball-based liquid animation, the metaballs' positions are the main parameters controlling the deformation of liquids. As how moving the metaballs to change a liquid from shape A to shape B is not within the scope of this research, it can be any movement-controlling scheme, such as in [8, 12, 13, 18]. Due to the above-mentioned constraint, our volume-preserving approach will not distort their animation effects.

Digital Object Identifier (DOI) 10.1007/s003710100164

All objects made of imcompressible substances, not only liquids, will preserve their volume when deformed. This drives many researchers to work on the volume-preserving problem. Sederberg and Parry pointed out in [21] 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 remains 1 while the users move control points is difficult. In [20], volume preservation is solved for objects defined by Bezier solids. Aubert and Bechmann presented a volume-preserving deformation method suitable for polyhedral objects in [2]. And Hirota and Maheshwari presented an efficient algorithm to preserve the total volume of a solid undergoing free-form deformation using a multi-level optimization [7]. But there are few approaches available for implicit surfaces. The only approach we have found is that presented in [15] by Murta and Miller. However, this method is inefficient regarding volume calculation and fails in preserving local volume.

The main obstacle for preserving the volume of liquids is that its implicit representation cannot provide an analytical way to calculate the volume. For this reason, the approach presented in this paper differs from the constrained minimization scheme used in most of the above-mentioned methods.

Due to the constraint of keeping the positions of metaballs unchanged, we select both the influence radius and the maximum density of every metaball as parameters to tune the volume because we need to adjust the two parameters to fit a wide variety of shape changes. Both of them are monotonic to volume, thereby simplifying the volume-adjusting scheme.

The speed of the volume calculation is essential in this approach. To efficiently calculate the volume, recursive subdivision is used. And the subdivision criterion, which is essential to the subdivision efficiency, is given by using the idea of interval analysis to cull away cells that do not straddle the implicit surface.

Another difficulty is that when the fluid consists of several isolated surfaces, we have to distinguish the metaballs generating one isolated surface from those of the others and preserve the volume enclosed by every isolated surface, because the volume variation of one isolated surface cannot propagate to the other isolated surfaces. This is achieved using an adjacency matrix, which is a concept in graph theory, and discussed in Sect. 4.

The proposed approach preserves the volume of liquids locally or globally, adjusting weights assigned to each metaball. That is to say, for those fluids with high viscosity, when deformation occurs only in a local part of the liquid and changes the volume, this approach will balance the volume to its original value within a local region near the deformation position, with the other parts of liquids far enough away remaining unchanged. It is accomplished by assigning a weight to each metaball to control its contribution to volume adjusting. For the local-volume preservation, the farther the metaball is from the deformation location, the smaller its weight, while for the global volume preservation, all the weights are set to the same value. The weights for the local-volume preservation are also generated by the adjacency matrix.

The remainder of this paper is organized as follows: Sect. 2 is an introduction of the volume-controlling parameters and their influence on volume variation, together with the main procedure for preserving volume; Sect. 3 proposes a volume-calculation scheme using recursive subdivision based on the idea of interval analysis; Sect. 4 describes how to localize the volume compensation; and following the examples in Sect. 5, concluding remarks are given in Sect. 6.

2 Volume-adjusting scheme

The surface of liquids generated by metaballs is defined by the points satisfying the following equation:

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

where T_0 is a threshold, q_i is a factor coefficient (also called the maximum density) of metaball I, and f_i is the density function of metaball I. The volume of the liquids can be described as $\iiint_{f(x,y,z)>0} \mathrm{d}x\mathrm{d}y\mathrm{d}z$.

Therefore, in the remainder of this paper, we will concentrate on the implicit surface $f(x, y, z) = 0$. The integral expression of volume cannot be computed analytically for most field functions, but it tells us that when the region in which $f(x, y, z) > 0$ is increased, the volume of the implicit surface also will increase.

There are many existing forms of metaball density functions that can be used in Eq. (1). The following are three typical forms proposed by Murakami,

Wyvill, and Nishimura respectively.

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

$$f_i(r) = -\frac{4}{9}\left(\frac{r}{R_i}\right)^6 + \frac{17}{9}\left(\frac{r}{R_i}\right)^4 - \frac{22}{9}\left(\frac{r}{R_i}\right)^2 + 1, \tag{3}$$

$$f_i(r) = \begin{cases} 1 - 3\left(\frac{r}{R_i}\right)^2 & (0 \le r \le \frac{R_i}{3}) \\ \frac{3}{2}\left(1 - \left(\frac{r}{R_i}\right)^2\right) & (\frac{R_i}{3} \le r \le R_i) \end{cases} \quad , \tag{4}$$

where r is the distance from point (x, y, z) to the center of metaball, R_i is the influence radius of metaball I. All have the 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 metaballs C^1 continuous.

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

From Eq. (1), we know that changing any of q_i, f_i or T_0 will change the region where $f(x, y, z) > 0$ and then change the volume. T_0 is monotonic to f, allowing for the simple control of volume. The problem is that it cannot meet the needs of local deformation. When T_0 changes, the entire implicit surface also will change. We thus have to fix T_0 and search for other volume control parameters.

Although the work of Desbrun et al. in [5] did not address the volume-preserving problem, they presented an approach to control the volume by adopting a new field function:

$$new f_i(r) = f_i(r - k), \tag{5}$$

where f_i is the traditional metaball field function as in Eqs. (2–4), r is the distance from a given point to the center of a metaball, and k is the variable used for controlling volume. We can increase the volume by assigning k a positive value, and decrease it by assigning k a negative one. The drawback is that when we want to decrease the volume and assign $k < 0$, the first derivative at $r = 0$ is not equal to zero ($new f_i'(0) \neq 0$), and this will cause the isosurface not to be smooth at some points.

Another existing approach is that of [15], in which Murta et al. 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 volume. And no matter how large q_i is, the isosurface cannot expand beyond its influence range. Therefore, sometimes we cannot adjust the current volume to its original value with the metaballs' influence radii unchanged. For example, when ten separated metaballs 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 assigned to q_i are, as the isosurface cannot expand beyond the ball with the influence radius.

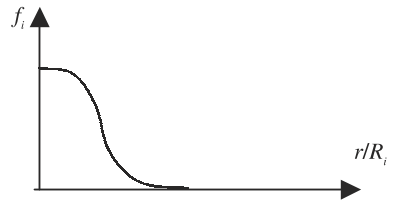


Fig. 1. Shape of field function

To overcome these disadvantages, we simultaneously use q_i and R_i as control parameters to adjust the volume. Both parameters are monotonic to the volume, so that 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 metaball-based object is as follows:

Step 1. Initialization, such as setting scene and time interval, and generating metaballs.

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

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

Step 3. Classify all the metaballs into different sets. This step is designed to handle the preserving of local volume. When a deformation occurs, the corresponding volume adjustment will be employed only for those metaballs that are in the same set as those participating in the deformation. A detailed scheme for the classification will be presented in Sect. 4.

Step 4. For each set J generated by step 3, calculate the volume V_{i-1}^J of an isosurface generated by metaballs in this set at time step t_{i-1}, then adjust the volume-controlling parameter q_k

and R_k of every metaball K in set J to make the current volume V_i^j equal to V_{i-1}^j. R_k and q_k are adjusted by $R_k = (1 + w_k * \mu) * R_k$ and $q_k = (1 + w_k * \mu) * q_k$ recursively, where w_k is a weight of the metaball K, which is used to control the metaball's contribution to volume variation. When controlling the volume, the larger w_k is, the more the volume-controlling parameters q_k and R_k will be changed. The detailed approach to set w_k will be introduced in Sect. 4. 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 5. Rendering.

In the above procedure, Step 1, Step 2, and Step 5 are common for a metaball-based animation, while Step 3 and Step 4 are particular to preserving volume.

3 Volume calculation using recursive subdivision

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. A pseudocode of calculating the volume of an object within region X using recursive subdivision can be described as follows:

Add X to stack S.
While S is nonempty
  Pop X from S
  If the size of X is smaller than a threshold EPS
    Add X to undetermined stack U
    Else if X is completely within the object
    Add X to within-object stack V
  Else if X is completely outside the object
    Discard X
  Else
    *Subdivide X into smaller region X*1, *X*2, . . . , *Xn,*
    And push them into S.
End while.

The sum of volumes of regions in within-object stack V plus half of volumes of regions in undetermined stack U is approximately regarded as the volume we are seeking.

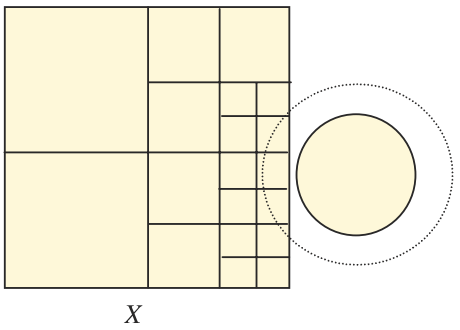


Fig. 2. An inaccurate criterion for subdivision

Although the recursive subdivision is more efficient than uniform sampling, the efficiency of this approach in practical use is determined by the accuracy and speed of judging whether X is inside or outside the object. Figure 2 illustrates an example of subdivision using the range of metaball influence as a judging criterion. Its inaccurate judgment causes unnecessary subdivision and lowers the efficiency. Interval analysis is a valid mathematic tool to help solve the inside–outside problem more accurately.

3.1 Interval analysis

An interval, $A = [a, b]$, is a subset of R defined as $[a, b] = \{x | a \le x \le b, x, a, b, \in R\}$, 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 (also called interval analysis [14, 22]) 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 $\{f(x)|(x \in X)\}$ be a traditional function, and the inclusion function for f, written as F, can be described as $F(X) = [F.lb, F.ub]$, where $F.lb \le \min_{x \in X} f(x)$, $F.ub \ge \max_{x \in X} f(x)$. 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

$$X+Y=[X.lb+Y.lb,\ X.ub+Y.ub], \tag{6}$$

$$X-Y=[X.lb-Y.ub,\ X.ub-Y.lb], \tag{7}$$

$$\begin{aligned} X*Y=[&\min(X.lb*Y.lb,\ X.lb*Y.ub,\ X.ub*Y.lb,\\ &X.ub*Y.ub),\ \max(X.lb*Y.lb,\ X.lb*Y.ub,\\ &X.ub*Y.lb,\ X.ub*Y.ub)], \end{aligned} \tag{8}$$

$$\begin{aligned} X/Y=[&\min(X.lb/Y.lb,\ X.lb/Y.ub,\ X.ub/Y.lb,\\ &X.ub/Y.ub),\ \max(X.lb/Y.lb,\ X.lb/Y.ub,\\ &X.ub/Y.lb,\ X.ub/Y.ub)]\quad 0\notin[Y.lb,\ Y.ub]\,, \end{aligned} \tag{9}$$

where $X=[X.lb,\ X.ub]$ and $Y=[Y.lb,\ Y.ub]$, and the proposed method does not use a division by intervals containing zero.
It is clear that the above interval operations (called formulae IA) can be applied recursively to yield an inclusion function for an arbitrary, nested combination of arithmetic operations. Thus, the inclusion function of all the polynomial functions can be evaluated by the combination of the above operations. It seems that this can be used in the recursive subdivision scheme to solve the inside–outside problem for the implicit surface generated by metaballs.
Actually, however, it is still not very practical, as the range intervals produced in this way are often much wider than the true ranges of the function. For example, for the function $f(x)=x(10-x),\ x\in[4,6]$, the true range of $f(x)$ is [24, 25], Applying the IA formulae, Eqs. (6–9), we have

$$10-X=[10,10]-[4,6]=[4,6]$$
$$X*(10-X)=[4,6]*[4,6]=[16,36].$$

This problem is particularly serious in long computation chains, where the intervals computed at one stage serve as inputs for the next [4].

3.2 Inclusion function for metaball field function

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 $X=[X.lb,X.ub]$, $Y=[Y.lb,Y.ub]$, $Z=[Z.lb,Z.ub]$, $k_i>0$.
Using formulae IA, we have $F(C)=[F.lb,\ F.ub]=\left[\sum_{i=1}^{N}k_iF_i.lb,\sum_{i=1}^{N}k_iF_i.ub\right]$, 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 the metaball 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, and thus the problem can be converted to finding $\max_{x\in X,y\in Y,z\in Z}r^2=\max_{x\in X,y\in Y,z\in Z}((x-x_i)^2+(y-y_i)^2+(z-z_i)^2)$ 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 center of metaball I. Because x,y,z are independent variables, we have

$$\begin{aligned} \max_{x\in X,y\in Y,z\in Z}r^2=&\max_{x\in X}(x-x_i)^2+\max_{y\in Y}(y-y_i)^2\\ &+\max_{z\in Z}(z-z_i)^2, \end{aligned} \tag{10}$$

$$\begin{aligned} \min_{x\in X,y\in Y,z\in Z}r^2=&\min_{x\in X}(x-x_i)^2+\min_{y\in Y}(y-y_i)^2\\ &+\min_{z\in Z}(z-z_i)^2. \end{aligned} \tag{11}$$

Now the problem has become simplified, with $\max_{x\in X}(x-x_i)^2$ and $\min_{x\in X}(x-x_i)^2$ obtained by the following codes:

```
if (x_i < (X.lb + X.ub)/2.0)
    max_{x∈X} (x − x_i)^2 = (X.ub − x_i)^2
  else
    max_{x∈X} (x − x_i)^2 = (X.lb − x_i)^2
if (x_i < X.lb)
    min_{x∈X} (x − x_i)^2 = (X.lb − x_i)^2
  else if (x_i > X.ub)
    min_{x∈X} (x − x_i)^2 = (X.ub − x_i)^2
  else
    min_{x∈X} (x − x_i)^2 = 0.0
```

Similarly, we can obtain $\max_{y\in Y}(y-y_i)^2$, $\max_{z\in Z}(z-z_i)^2$ and $\min_{y\in Y}(y-y_i)^2$, $\min_{z\in Z}(z-z_i)^2$, and then use Eqs. (10) and (11) to obtain $\max_{x\in X,y\in Y,z\in Z}r^2$ and $\min_{x\in X,y\in Y,z\in Z}r^2$.

In this way, the computation cost of calculating the range of field function $f(x,y,z)$ in a given cube is just twice that of calculating $f(x,y,z)$ at one point. And the range calculated in this way is much more accurate than using traditional interval analysis [22], affine arithmetic [4], or Lipschitz condition [6].

4 Local-volume preserving

4.1 Metaball classification for volume preserving

Volume preserving can be classified into the preserving of global volume and the preserving of local volume. The former means preserving the entire volume of the object, and the latter means preserving all volumes of the defined object areas.

Our goal is to preserve the local volume, namely, to limit the volume compensation within the region near the place where volume variation occurs. This step is necessary to meet the demands of local deformation. Particularly for those objects consisting of several isolated surfaces, it is simply irrational to adjust all the surfaces to make up for the volume variation caused by the deformation of a single surface. As illustrated in Fig. 3, five metaballs V_1, V_2, V_3, V_4 and V_5 generate two isolated surfaces S_1 and S_2. V_4 is moving and V_1, V_2, V_3, V_5 are motionless. The volume of S_1 varies because the movement of V_4, and S_2 remains unchanged. In this case, it is unreasonable to adjust all the metaballs' influence radii and maximum densities to compensate for the volume variation of S_1, because the volume variation cannot propagate from S_1 to another isolated surface S_2. We therefore must distinguish the metaballs generating S_1 from the others, and make the volume-adjusting scheme applicable only to those generating S_1.

To achieve this goal, we must first obtain the following information at each time step t_i:

- The number of the isolated surfaces
- How many metaballs compose each isolated surface, and which are they

We can solve this problem with the help of graph theory.

Assuming the number of metaballs is N, all the metaballs (denoted by V_i $(i = 1, 2, \ldots, N)$) are regarded as vertices of graph$G = (V, E)$, with the edge of the graph defined as:

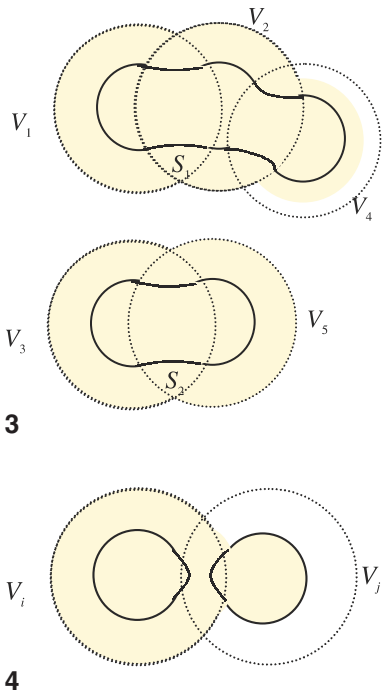


Fig. 3. Two insolated surfaces generated by metaballs

Fig. 4. Two metaballs having overlapped influence ranges can generate two insolated surfaces

If V_i and V_j satisfy Condition I, there is an edge $\{V_i, V_j\}$ between V_i and V_j, otherwise there is no edge between them.

Condition I: *V_i and V_j have overlapped influence range. Influence range of a metaball refers to the ball with the position of the metaball as its center, and the influence radius of metaball as it radius.*

An informal explanation for Condition I is that metaball V_i and V_j are connected directly, as are V_1 and V_2 in Fig. 3. In fact it is not sufficient for Condition I to judge that V_i and V_j are connected directly, on the grounds that they can generate two isolated surfaces although their influence ranges are overlapped, as indicated in Fig. 4. But in this case, when we adjust the influence radius of V_i, the volume of V_j will change correspondingly. We must, therefore, consider them as a whole when we adjust their volume-controlling parameters, because they have influences on each other's volume, although they actually are not in the same isolated surface. Applying Condition I to Fig. 3, we find that there are edges between V_1

and V_2, V_2 and V_4, V_3 and V_5, but no edge between V_1 and V_3, V_1 and V_4, V_1 and V_5, V_2 and V_3, V_2 and V_5, V_3 and V_4, V_4 and V_5. According to this definition, Fig. 3 can be described as an undirected graph, as shown in Fig. 5.

In the graph representing the metaballs and their relationships, that metaballs V_i and V_j are within the same isolated surface can be interpreted as meaning there is at least one path between vertex V_i and V_j. And the connected components of the graph refer to the isolated surfaces in space. The information we want can then be described as:

- The number of connected components of the graph
- The vertices of each connected component

This can be obtained by using an adjacency matrix. Suppose that the vertices of $G(V, E)$ are listed arbitrarily as $V_1, V_2, V_3, \ldots, V_n$. The adjacency matrix of G, with respect to this listing of the vertices, is the $n \times n$ zero–one matrix with 1 as its (i, j)th entry when V_i and V_j are adjacent, and 0 as its (i, j)th entry when they are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1, & \text{if } \{V_i, V_j\} \text{ is an edge of } G \\ 0, & \text{else} \end{cases}$$

Using the adjacency matrix, we can define a connection matrix of G as $C = I + A + A^2 + \cdots + A^{n-1}$, where I is the identity matrix. The connection matrix $C = [c_{ij}]$ has the property that

$$\begin{cases} \text{There is at least one path between vertex } V_i \text{ and } V_j, & \text{if } c_{ij} \neq 0 \\ \text{There is no path between vertex } V_i \text{ and } V_j, & \text{if } c_{ij} = 0 \end{cases}$$

After exchanging several rows and columns, the connection matrix can be rewritten in the form of the block matrix $\begin{bmatrix} C_1 & 0 & 0 & \cdots & 0 \\ 0 & C_2 & 0 & \cdots & 0 \\ 0 & 0 & C_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & C_k \end{bmatrix}$, where C_i is a submatrix with nonzero components. Each C_i represents one connected component of graph G. So the number of submatrices C_i equates to the number of isolated surfaces S_i, the vertices in C_i referring to the metaballs generating S_i. For example, the adjacency matrix of graph in Fig. 3 is

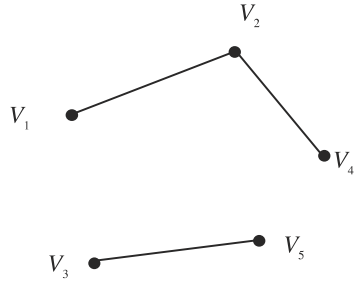


Fig. 5. The corresponding undirected graph of Fig. 3

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array},$$ and its corresponding connection matrix is $$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 4 & 3 & 0 & 3 & 0 \\ 3 & 7 & 0 & 3 & 0 \\ 0 & 0 & 3 & 0 & 2 \\ 3 & 3 & 0 & 4 & 0 \\ 0 & 0 & 2 & 0 & 3 \end{bmatrix} \end{array}.$$ By exchanging the third and fourth rows and columns respectively, we have $\begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_4 \\ v_3 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_4 & V_3 & V_5 \end{array} \\ \begin{bmatrix} 4 & 3 & 3 & 0 & 0 \\ 3 & 7 & 3 & 0 & 0 \\ 3 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 2 & 3 \end{bmatrix} \end{array}.$ Now we can draw the conclusion that there are two isolated surfaces S_1 and S_2. S_1 is composed of metaballs V_1, V_2,and V_4, while S_2 is composed of V_3 and V_5.

Thus far, we can classify the metaballs into several sets W_j at each time step t_i. Each set W_j contains metaballs generating the isolated surface S_j. But there still exists a serious problem: the sets evaluated at time step t_i may be different from those at time step t_{i-1}. That is to say, the number of isolated surfaces at time step t_i and t_{i-1} might be different, and the metaballs generating them different, too. For example, assume Fig. 3 is the scene at time step t_{i-1}. At time step t_i, metaball V_4 moves to a new location, illustrated in Fig. 6a. The result of classification at time step t_{i-1} is $\left\{W_1^{i-1}, W_2^{i-1}\right\} = \{\{V_1, V_2, V_4\}, \{V_3, V_5\}\}$, and that at time step t_i is $\left\{W_1^i, W_2^i, W_3^i\right\} = \{\{V_1, V_2\}, \{V_3, V_5\}, \{V_4\}\}$. In this

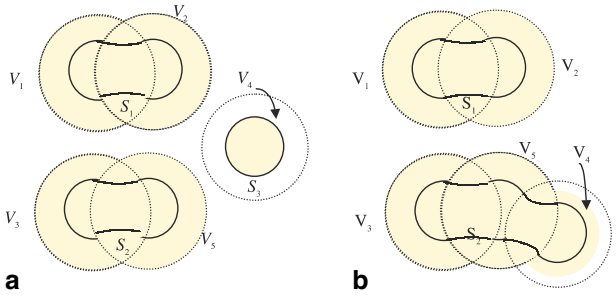


Fig. 6. Metaball V_4 moves to a new location at time step t_i

case, what we should do is to adjust the volume within $\{V_1, V_2, V_4\}$ and $\{V_3, V_5\}$. If metaball V_4 moves to a new location, as in Fig. 6b, the classified sets at time step t_i then become $\left\{W_1^i, W_2^i\right\} = \{\{V_1, V_2\}, \{V_3, V_4, V_5\}\}$. This time we should adjust V_1,V_2,V_3, V_4,V_5 simultaneously to preserve volume, because the volume variation can propagate from isolated surface S_1 to S_2 through the movement of V_4. Thus, when the metaballs' classifications $\left\{W_1^{i-1}, W_2^{i-1}, \ldots, W_{k_{i-1}}^{i-1}\right\}$ at time step t_{i-1} and $\left\{W_1^i, W_2^i, \ldots, W_{k_i}^i\right\}$ at time step t_i are different, we must create a new classification accordingly.

Actually, for a set containing all the metaballs $W = \{V_1, V_2, V_3, \ldots, V_n\}$, the metaball classification $P = \left\{W_1^i, W_2^i, \ldots, W_{k_i}^i\right\}$ is a partition of W. When there are two different partitions $P_{i-1} = \left\{W_1^{i-1}, W_2^{i-1}, \cdots, W_{k_{i-1}}^{i-1}\right\}$ and $P_i = \left\{W_1^i, W_2^i, \cdots, W_{k_i}^i\right\}$ at successive time steps t_{i-1} and t_i, the new partition P on which we need to preserve the volume can be generated by a union operation of partition P_{i-1} and P_i. The detailed explanation of the union operation of two partitions can be found in the Appendix.

4.2 Weight of metaball for adjusting volume

Now that we have created a partition $P = \left\{W_1, W_2, \cdots, W_k\right\}$ to preserve the volume within each W_j, we can concentrate on the volume variation and adjustment within one isolated surface. Different substances will reflect volume variation differently within the object. For example, for fluids, volume variation will propagate over the entire isolated surface, but for an elastic object, the compensation of volume variation will be limited to a region near the location of the volume variation itself, the scope of the region depending on the flexibility of the substance. To describe this difference, we provide a sequence of parameters to control the scope of compensation region corresponding to the flexibility of substances.

According to the distance from the location of volume variation, we assign a weight w_k to each metaball K, and take $R_k = (1 + w_k{*}Rstep) * R_k$, $q_k = (1 + w_k{*}qstep) * q_k$ to adjust the influence radius and maximum density of metaball K to balance the volume variation. This means the larger the weight w_k, the more the volume-controlling parameters will change. And if $w_k = 0$, the volume-controlling parameters of metaball K remain unchanged, meaning that the local shape near metaball K is preserved. We therefore set a larger weight to the metaball near the location of volume variation and a smaller weight to the more distant metaball.

When the influence radius, maximum density and threshold T_0 remain unchanged, the main cause of volume variation is the movement of metaballs. We then might take the location of moving metaballs as the place where volume variation occurs, and assign a large weight to the moving metaballs and those directly connected. In the related graph, the vertex V_i corresponding to the moving metaball and those vertices adjacent to V_i should be assigned a large weight, and those whose shortest path to V_i is a considerable distance should be assigned a small weight. To achieve this, we can make use of the adjacency matrix $A = [a_{ij}]$. Assume $A^s = \left[a_{ij}^{[s]}\right]$, we can easily understand that $a_{ij}^{[2]} \neq 0$ means there is at least one path of length 2 from V_i to V_j since $a_{ij} \neq 0$ means vertices V_i and V_j are adjacent. And the value of $a_{ij}^{[2]}$ refers to the number of paths of length 2 from V_i toV_j. Similarly the (i, j)th entry of matrix A^s means the number of paths of length s from V_i to V_j. With this property of adjacency matrix, we can make a slight adjustment to the connection matrix to generate a matrix $G = [g_{ij}]$ for weight assignment:

$$G = c_0 I + c_1 A + c_2 A^2 + \cdots + c_{N-1} A^{N-1}$$

Where c_s are coefficients satisfying $c_0 = c_1 > c_2 > c_3 > \ldots > c_{N-1}$, which refers to the volume-compensation duty that a metaball having a path of length s to a moving metaball should undertake. Therefore, if V_i is moving, the volume-compensation duty of metaball V_j to the volume variation caused by V_i can be represented as $g_{ij} = \sum_{s=0}^{N-1} c_i a_{ij}^{[s]}$. And

$d_j = \sum_{i \in M} g_{ij} = \sum_{i \in M} \sum_{s=0}^{N-1} c_s a_{ij}^{[s]}$ means the duty of V_j to all the volume variation in the isolated surface, where M is the set of moving metaballs. Finally, we assign $w_j = \frac{d_j}{\sum_j^{N-1} d_j}$ as the weight of metaball V_j. The coefficient c_s, therefore, represents in essence the basic parameters to distribute the volume-compensation duty to each metaball. A corresponding scope of volume-compensation region suited to a given substance can be roughly determined by manually assigning c_s a proper value. For example, $[c_0, c_1, c_2, c_3, c_4, c_5, \ldots, c_{N-1}] = [4, 4, 2, 1, 0, 0, \ldots, 0]$ will make the volume-adjusting scheme applicable only to those metaballs whose shortest path to the moving metaball is shorter than length 4. And within these volume-compensation regions, all metaballs can be classified into four types according to the length of their shortest path to the moving metaballs:

- Length-0 type (moving metaballs)
- Length-1 type (metaballs directly connected with the moving metaballs)
- Length-2 type (metaballs having the shortest path of length 2 to the moving metaballs)
- Length-3 type (metaballs having the shortest path of length 3 to the moving metaballs)

The proportion among volume-compensation duty of these four types is about 4 : 4 : 2 : 1.
And to the substance of fluids, $[c_0, c_1, c_2, c_3, c_4, c_5, \ldots, c_{N-1}] = [1, 1, 1, 1, 1, 1, \ldots, 1]$ is applicable, which means that all the metaballs take part in the volume-adjusting scheme.

5 Examples

We have applied this approach to simulate drops of water falling from a horizontal plane. The original scene is composed of 38 metaballs on the same horizontal plane. The maximum densities q_i's 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. Figures 7 and 8 simulate the process with different parameters. In Fig. 7, the weights of all the metaballs are assigned the same value, allowing the part remaining on the plane to shrink evenly. In the end scene of Fig. 7 (the Fig. 7c), the maximum densities of these metaballs are all equal to 0.94. In Fig. 8, the volume is preserved locally.

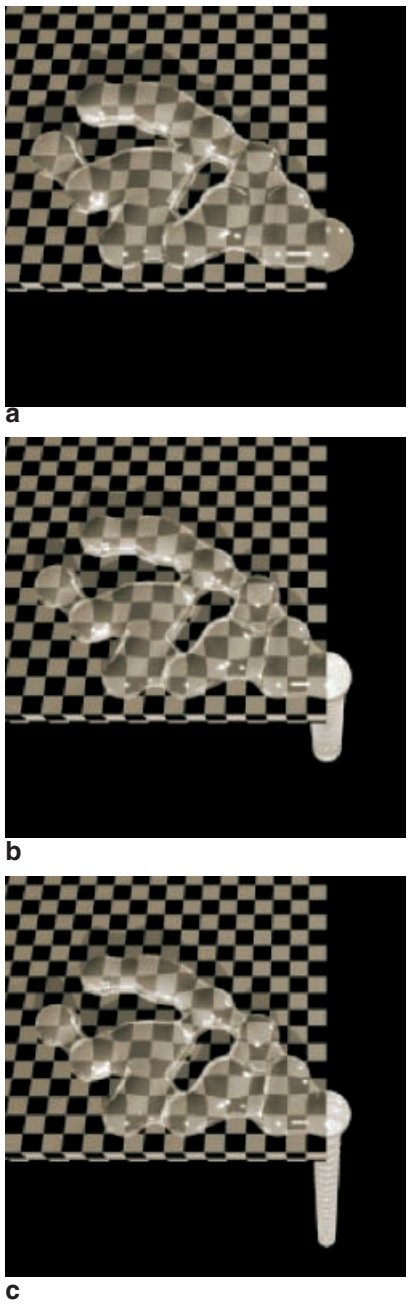


Fig. 7. Global volume-preserving

The controlling sequence c_s is specified as the values $[4, 4, 2, 0.2, 0, \ldots, 0]$, so that the region near the location where drops fall shrinks more, and the region sufficiently removed from that same loca-

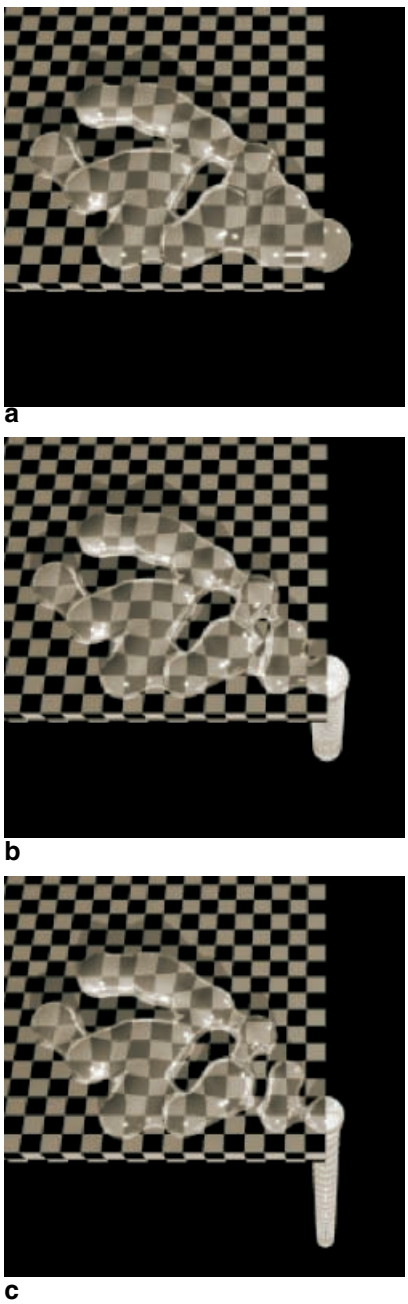


Fig. 8. Local volume-preserving

tion remains unchanged. The maximum densities of metaballs in Fig. 8c range from 0.77 to 1.0, according to the distance from the location of the falling drops.

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

6 Conclusions

In this paper, we have presented a volume-preserving approach for liquids modeled by metaballs. We adjust the volume of liquids to an original volume by tuning the influence radius and maximum density of metaballs, thereby avoiding distortion of the animation effect, since in most animation, the deformation of liquids is achieved by the movement of metaballs.
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 metaball density function. We can know from Sect. 3.2 that we obtained the range of density function within the entire cube only by the computation cost of calculating the function at two points. This is much better than using traditional interval analysis or affine arithmetic [4], both in speed and accuracy. It is also more advanced than the Lipschitz condition approach [6] because our approach can quickly calculate the accurate range of density function generated by a single metaball in a cube, while the Lipschitz condition approach introduced an expansion of the real range by approximating $|f(x_1) - f(x_2)|$ with $\lambda \|x_1 - x_2\|$, where λ is the upper bound of the derivative of $f(x)$ within $[x_1, x_2]$.
Graph theory enables us to distinguish the metaballs generating one isolated surface from those generating another, and makes the volume-adjusting scheme applicable only to those metaballs belonging to the isolated surface whose volume varies. Furthermore, the scope of the volume-compensation region can be controlled so that the volume can be preserved locally.
We used Eq. (2) as a density function, but the approach can be applied to any metaball field function having the shape of Fig. 1. This approach is applicable to all objects, not only liquids, generated by metaballs and animated by metaball movement. To increase versatility, a generalization to make the approach suitable for all skeleton-based implicit surfaces is expected to result from further research.

References

1. Ashraf G, Wong KC (1999) Dust and water splashing models for hopping figures. J Vis Comput Anim 10(4):193–213

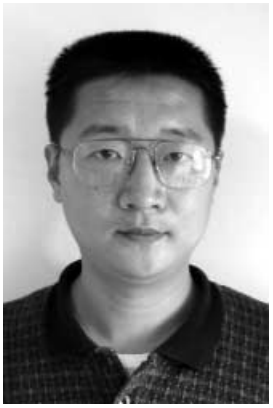
2. Aubert F, Bechmann D (1997) Volume-preserving space deformation. Comput Graph 21(5):625–637
3. Blinn J (1982) A generalization of algebraic surface drawing. ACM Trans Graph 1(3):235–256
4. Comba JLD, Stolfi J (1993) Affine arithmetic and its applications to computer graphics. In: Proceedings of the VI Sibgrapi, pp 9–18
5. Desbrun M, Gascuel MP (1995) Animating soft substances with implicit surfaces. In: Proceedings of SIGGRAPH '95, pp 287–290
6. Galin E, Akkouche S (2000) Incremental polygonization of implicit surfaces. Graph Models 62(1):19–39
7. Hirota G, Maheshwari R, Lin MC (2000) Fast volume-preserving free-form deformation using multi-level optimization. CAD 32(8/9):499–512
8. Kaneda K, Ikeda S, Yamashita H (1999) Animation of water droplets moving down a surface. J Vis Comput Anim 10(1):15–26
9. Kass M, Miller G (1990) Rapid, stable fluid dynamics for computer graphics. Comput Graph 24(4):49–57
10. Foster N, Metaxas D (1996) Realistic animation of liquids. Graph Models Image Process 58(5):471–483
11. Fournier A, Reeves B (1986) A simple model of ocean waves. In: Proceedings of SIGGRAPH '86, pp 75–84
12. Fournier P, Habibi A, Poulin P (1998) Simulating the flow of liquid droplets. In: Proceedings graphics interface 98, pp 133–142
13. Fujita T, Hirota K, Murakami K (1990) Representation of splashing water using metaball model. Fujitsu 41(2):159–165
14. Moore RE (1966) Interval analysis. Prentice Hall, Englewood Cliffs, New Jersey
15. Murta A, Miller J (1999) Modelling and rendering liquids in motion. In: WSCG'99 Proceedings, pp 194–201
16. Nishimura H, Hirai M, Kawai T, Kawata T, Shirakara I, Omura K (1985) Object modeling by distribution functions. Electron Commun 68D(4):718–725
17. Nishita T, Nakamae E (1994): Method of displaying optical effects within water: Using the Accumulation Buffer. In: Proceedings of SIGGRAPH '94, pp 24–29
18. O'Brien JF, Hodgins JK (1995) Dynamic simulation of splashing fluids. In: Proceedings of *Computer Animation '95*, Geneva Switzerland, April 19-21, pp 198–205
19. Peachy D (1986) Modeling waves and surf. In: Proceedings of SIGGRAPH '86, pp 65–74
20. Rappoport A, Sheffer A, Bercovier M (1996) Volume preserving free-form solids. IEEE Trans Vis Comput Graph. 2(1):19–27
21. Sederberg TW, Parry SR (1986) Free-form deformation of solid geometric models. Comput Graph 20(4):151–160
22. Snyder JM (1992) Interval analysis for computer graphics. Comput Graph 26(2):121–130
23. Yu YJ, Jung HY, Cho HG (1999) A new water droplet model using metaball in the gravitational field. Comput Graph 23(2):213–222

Appendix
Procedure of union operation of partition

Assume $P_{i-1} = \left\{ W_1^{i-1}, W_2^{i-1}, \ldots, W_{k_{i-1}}^{i-1} \right\}$ and $P_i = \left\{ W_1^i, W_2^i, \ldots, W_{k_i}^i \right\}$ are two different partitions of set $W = \{V_1, V_2, V_3, \ldots, V_n\}$, the union operation of P_{i-1} and P_i can be described by the following pseudocode.

$P'_{i-1} = P_{i-1}$; $P'_i = P_i$; $N = 0$; $P = \Phi$;
    */*N: number of the blocks of the new partition*/*
While $(P'_{i-1} \neq \Phi)$*{*
  $N++$;
  $W_N =$ *any element of* P'_{i-1};
  While (there is at least one element W_j^i *in* P'_i*, satisfy* $W_j^i \cap W_N \neq \Phi$*) {*
    $W_N = W_N \cup W_j^i$;
    $P'_i = P'_i - \left\{ W_j^i \right\}$;
    While (there is at least one element W_k^{i-1} *in* P'_{i-1}*,*
    satisfy $W_k^{i-1} \cap W_N \neq \Phi$*) {*
      $W_N = W_N \cup W_k^{i-1}$;
      $P'_{i-1} = P'_{i-1} - \left\{ W_k^{i-1} \right\}$;
      }
    }
  $P = P + \{W_N\}$
  }
$P = \{W_1, W_2, W_3, \cdots, W_N\}$ *is the new partition we seek.*

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



RUOFENG TONG is an associate professor in the Department of Computer Science, Zhejiang University, China. He was a visiting researcher in the Faculty of Engineering, Hiroshima University, Japan, from 1999 to 2001. He received his BE from Fudan University, in 1991, and a PhD from Zhejiang University China in 1996. His research interests include geometric modeling, computer graphics and animation.

KAZUFUMI KANEDA is an associate professor in the Department of Information Engineering at Hiroshima University. He worked at the Chugoku Electric Power Company, Japan, from 1984 to 1986. He joined Hiroshima University in 1986. He was a visiting researcher in the Engineering Computer Graphics Laboratory at Brigham Young University in 1991. Kaneda received his BE, ME, and DE in 1982, 1984, and 1991, respectively, from Hiroshima University. His research interests include computer graphics, scientific visualization, and image processing. He is a member of the ACM, the IPSJ, the IEICE, and the IEEJ.

HIDEO YAMASHITA is a Professor in the Department of Information Engineering at Hiroshima University. He received his BE and ME in electrical engineering from Hiroshima University, in 1964 and 1968, respectively, and a PhD in electrical engineering from Waseda University, Tokyo, in 1977. His interests include electric and magnetic fields analysis of electric machinery by finite-element analysis and visualization of 3D magnetic fields by computer graphics. He is a member of the IEEJ and the IEEE.