Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections

Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro & Tomoyuki Nishita

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789

Vis Comput
DOI 10.1007/s00371-014-1055-x

Springer

Vis Comput
DOI 10.1007/s00371-014-1055-x

ORIGINAL ARTICLE

Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections

Tetsuya Takahashi · Yoshinori Dobashi · Issei Fujishiro · Tomoyuki Nishita

Abstract We propose a particle-based hybrid method for simulating volume preserving viscoelastic fluids with large deformations. Our method combines smoothed particle hydrodynamics (SPH) and position-based dynamics (PBD) to approximate the dynamics of viscoelastic fluids. While preserving their volumes using SPH, we exploit an idea of PBD and correct particle velocities for viscoelastic effects not to negatively affect volume preservation of materials. To correct particle velocities and simulate viscoelastic fluids, we use connections between particles which are adaptively generated and deleted based on the positional relations of the particles. Additionally, we weaken the effect of velocity corrections to address plastic deformations of materials. For one-way and two-way fluid-solid coupling, we incorporate solid boundary particles into our algorithm. Several examples demonstrate that our hybrid method can sufficiently preserve fluid volumes and robustly and plausibly generate a variety of viscoelastic behaviors, such as splitting and merging, large deformations, and Barus effect.

Keywords Fluid simulation · Viscoelasticity · Deformation · Volume preservation · Velocity correction

T. Takahashi (✉)
UEI Research, Keio University, Tokyo, Japan
e-mail: tetsuya.takahashi@uei.co.jp

Y. Dobashi
Hokkaido University, UEI Research, Sapporo, Japan
e-mail: doba@ime.ist.hokudai.ac.jp

I. Fujishiro
Keio University, Tokyo, Japan
e-mail: fuji@ics.keio.ac.jp

T. Nishita
UEI Research, Hiroshima Shudo University, Tokyo, Japan
e-mail: tomoyuki.nishita@uei.co.jp

1 Introduction

Particle-based methods have been popular techniques to simulate fluids, rigid bodies, deformable objects, and their interactions. These methods use a simple concept of particle interactions, which can be easily extended to simulate various types of objects, and were adopted for a versatile simulation framework [16]. For conceptual simplicity and versatility, we herein focus on particle-based methods.

Since viscoelastic materials play an important role in representing common materials (e.g., egg white, gels, toothpaste, and slime) and producing visually attractive characters and effects, simulating viscoelastic materials has been required for movies and video games. In such entertainments, exaggerated representations which real materials do not exhibit, e.g., very large deformations, are frequently required to make characters and effects appear interesting and impressive.

In computer graphics, particle-based methods for simulating viscoelastic fluids have been proposed. However, previous particle-based, smoothed particle hydrodynamics (SPH) methods which depend on physical-based viscoelastic models [6,17] cannot simulate highly deformable objects because, at a particle position, physical contributions from farther particles are likely to be smaller owing to the decreasing property of finite support kernels, and consequently, viscoelastic materials fail to restore their original shapes when they undergo significant deformations. This problem can be addressed by combining SPH and a geometric method [7,28]. However, these hybrid methods have another problem that they suffer from volume loss of materials, which leads to unphysical fluid behaviors, because the geometric methods adopted by Takamatsu and Kanai [28] and Clavet et al. [7] make an incompressible fluid solver fail to preserve fluid volumes.

Published online: 03 December 2014

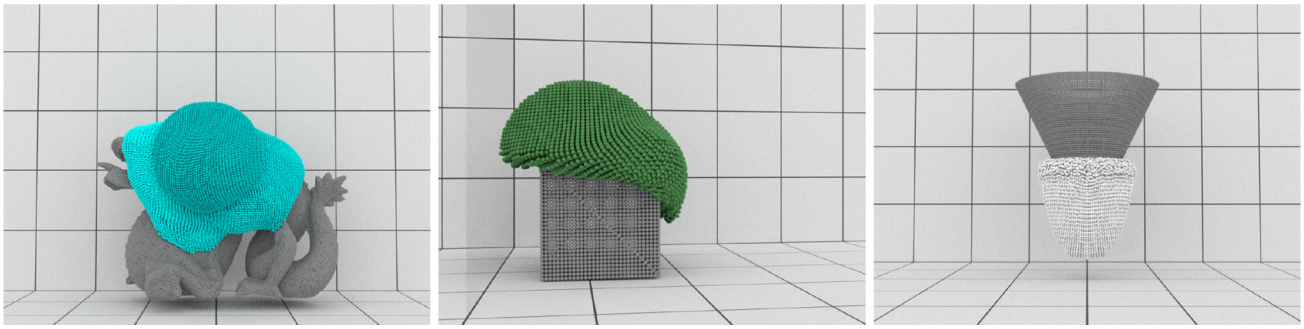

Fig. 1 Viscoelastic fluids simulated with our hybrid method. (*Left*) Merging spheres with different viscoelasticity values colliding with a static solid dragon. (*Middle*) A viscoelastic ball colliding with a solid cube. (*Right*) Barus effect of a viscoelastic material (see Sect. 4.4)

To solve the two problems above simultaneously, we propose a particle-based hybrid method for simulating viscoelastic fluids, which can preserve fluid volumes and enable large deformations of the fluids. Our method uses a combination of SPH and position-based dynamics (PBD) [4,20]. While using SPH to preserve fluid volumes, we exploit PBD for velocity corrections which achieve viscoelastic effects involving large deformations without negatively influencing the volume preservation. To correct particle velocities, we use a set of connections between particles, which are adaptively updated based on the positional relations of the particles. We incorporate solid boundary particles into our algorithm to simulate adhesion of viscoelastic fluids to solid objects.

Our key contribution lies in velocity corrections which allow for combining a geometric method with SPH to simulate viscoelastic behavior with large deformations while preserving fluid volumes, since the SPH-based methods [6,17] cannot handle large deformations and the previously proposed hybrid methods [7,28] cannot preserve the volume of viscoelastic fluids. Figure 1 illustrates characteristic behaviors of viscoelastic materials simulated with our hybrid method.

This paper is an extended version of our previous paper [26]. In this paper, we focus on the position-based velocity correction scheme. As new additional materials, we include a comparison of our method with the SPH-based method [17] and propose a boundary-handling method for fluid-solid coupling.

2 Related work

Viscoelastic fluids Miller and Pearce [18] and Terzopoulos et al. [30] proposed a spring-based model that computes repulsion and attraction forces among particles to simulate viscoelastic materials. Their method was adopted by Steele et al. [25] and Tamura et al. [29]. Clavet et al. [7] combined this spring-based method with SPH to simulate materials with elasticity, plasticity, and viscosity, adopting a prediction–relaxation scheme. A similar spring-based method was also proposed by Takahashi et al. [27], who used PBD to simulate fluids with viscosity and elasticity in a unified framework. Takamatsu and Kanai [28] combined shape matching (SM), which was originally proposed by Müller et al. [21] to simulate deformable objects, with SPH to fast and robustly simulate viscoelastic fluids.

Müller et al. [22] proposed an elasticity term which uses moving least square (MLS) to simulate elastoplastic objects. Solenthaler et al. [24] adopted the formulation of this elasticity term and computed it using SPH instead of MLS to allow for robustly simulating fluid with various properties under the condition of collinear or coplanar particle distributions. The method of Solenthaler et al. [24] was extended to handle rotational motions of elastic materials [2]. Mao and Yang [17] introduced a viscoelastic force term, called nonlinear corotational Maxwell model, into the Navier–Stokes equations to simulate viscoelastic fluids. A method similar to the method of Mao and Yang [17] was proposed by Chang et al. [6].

Gerszewski et al. [9] proposed a new formulation for simulating elastoplastic materials, which uses affine transformations to compute the gradient of deformations. This formulation was solved by Zhou et al. [31] in an implicit manner to efficiently and robustly perform simulations with larger time steps. The method of Gerszewski et al. [9] was extended by Jones et al. [14]. Jones et al. [14] used MLS to robustly simulate elastoplastic materials with spatially and temporally varying masses.

Volume preservation In particle-based simulations, volume preservation has been achieved by minimizing density fluctuations from the fluid rest density by enforcing fluid incompressibility [13] since meshes which are often used for volume preservation of deformable objects are inappropriate because of frequent topology changes. After Müller et al. [19] presented the basic SPH method, which suffers from unacceptable density fluctuations and volume loss because of the dependence on an equation of state (EOS), Becker and

Teschner [3] proposed weakly compressible SPH and alleviated large density fluctuations by replacing the EOF in [19] with a stiffer EOS called Tait equation, requiring exceedingly smaller time steps for stable simulation. Solenthaler et al. [23] proposed a predictive-corrective scheme, called predictive-corrective incompressible SPH (PCISPH) to use larger time steps while satisfying fluid incompressibility. A similar predictive-corrective scheme called local Poisson SPH was proposed by He et al. [10]. Bodin et al. [5] proposed a system of velocity constraints to improve the accuracy of PCISPH [23], while achieving incompressible flows. To further accelerate enforcing fluid incompressibility by allowing us to use larger time steps than those used in PCISPH [23], Macklin and Müller [15] adapted PBD for fluid simulation, and Ihmsen et al. [12] proposed implicit incompressible SPH (IISPH), which computes pressure values using semi-implicit integration. A method that enforces fluid incompressibility using semi-implicit integration, called incompressible SPH (ISPH) was also proposed by Cummins and Rudman [8].

3 Proposed method

The key of our method is to separately achieve volume preservation and large deformations of viscoelastic materials. We rely on a particle-based incompressible fluid solver to preserve fluid volumes while exploiting an idea of PBD [4,20] to correct particle velocities for viscoelastic effects without negatively affecting convergence of density fluctuations in the fluid solver.

We briefly describe computational process and important features of particle-based incompressible fluid solvers in Sect. 3.1 and explain our velocity correction scheme, which enables large deformations of viscoelastic materials while preserving their volumes in Sect. 3.2. In Sect. 3.3, we summarize our algorithm, giving a list of simulation procedures and implementation details.

3.1 Particle-based incompressible fluid solvers

To preserve fluid volumes, we use a particle-based incompressible fluid solver, such as ISPH [8], PCISPH [23], and IISPH [12]. In these SPH-based incompressible fluid solvers, fluids are discretized by particles (particle i has position $\mathbf{x}_i$, velocity $\mathbf{v}_i$, and mass m_i), and pressure p_i is iteratively corrected to obtain pressure force $\mathbf{F}_i^p$ which achieves small deviations of density ρ_i to the rest density ρ_0 less than a certain criterion, e.g., 0.01 %. We briefly explain the flow of computations for the solvers. For detailed simulation procedures and discussions, refer to the papers of particle-based incompressible fluid solvers [8,12,23].

We first compute intermediate velocity $\mathbf{v}_i^*$ for particle i by

$$\mathbf{v}_i^* = \mathbf{v}_i^t + \Delta t \frac{\mathbf{F}_i^{\text{other}}}{m_i}, \tag{1}$$

where t denotes time, Δt time step, and $\mathbf{F}^{\text{other}}$ all forces except pressure and viscoelastic ones. Next, we use an incompressible fluid solver to obtain pressure p_i, which is clamped as $p_i = 0$ if $p_i < 0$ not to cause attraction forces between particles, through iterative corrections with intermediate velocity $\mathbf{v}_i^*$ and compute pressure force $\mathbf{F}_i^p$ by

$$\mathbf{F}_i^p = -m_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}, \tag{2}$$

where j denotes a neighboring fluid particle and W_{ij} a kernel with a kernel radius h. Then, we compute velocity $\mathbf{v}_i$ for incompressible flows:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^* + \Delta t \frac{\mathbf{F}^p}{m_i}. \tag{3}$$

Finally, we update particle position $\mathbf{x}_i$:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i. \tag{4}$$

There are two important requirements to note; first, particle velocity $\mathbf{v}_i$ needs to be updated with pressure force $\mathbf{F}_i^p$ with Eq. (3) to integrate particle position $\mathbf{x}_i$ with Eq. (4), achieving incompressible flows. However, the method of Takamatsu and Kanai [28] does not satisfy this requirement because their method mixes velocities derived from SM, which does not ensure incompressible flows, and SPH, and thus resulting velocities cannot achieve incompressible flows. Second, particle position $\mathbf{x}_i$ must be fixed after neighbor search steps until pressure force $\mathbf{F}_i^p$ is computed with Eq. (2). As explained in [12,23], neglecting changes of particle positions causes erroneous sets of neighboring particles and their interparticle distances and leads to inaccurate estimates of physical quantities, which can make the fluid solver fail to converge or delay the convergence of the iterations. Since the method of Clavet et al. [7] uses a prediction-relaxation scheme, which change particle positions, this requirement is not satisfied.

3.2 Velocity correction scheme

Satisfying the two requirements mentioned above, we correct particle velocities to achieve viscoelastic effects with velocity correction vector $\Delta \mathbf{v}_i$ (see Sect. 3.2.1). We obtain intermediate velocity $\mathbf{v}_i^*$ by modifying Eq. (1):

$$\mathbf{v}_i^* = \mathbf{v}_i^t + \Delta t \frac{\mathbf{F}_i^{\text{other}}}{m_i} + \Delta \mathbf{v}_i. \tag{5}$$

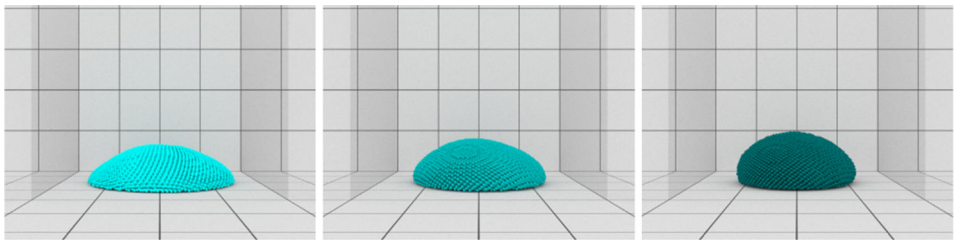

Fig. 2 A ball with different viscoelasticity values dropped onto the ground. Particles are color coded based on their velocity correction coefficients c (low *cyan* and high *dark cyan*)

In [26], although three velocity correction schemes (shape matching, spring-based, and position-based) were proposed to compute velocity correction vector $\Delta\mathbf{v}$, we herein use the position-based scheme because of the following advantages over the other schemes; first, the position-based scheme requires fewer neighbor particles than the shape matching scheme and uses pairwise connections which can simplify boundary conditions with other objects unlike the shape matching scheme which needs to construct particle clusters including farther particles. Second, the position-based scheme inherits properties of PBD [4,20] and thus is robust and easy to tune parameters as compared to the spring-based scheme.

3.2.1 Position-based velocity correction

To compute velocity correction vector $\Delta\mathbf{v}$, we construct a set of pairwise connections for neighboring particles when an object is created, with their initial interparticle distance r_{ij}, which is initialized as $r_{ij} = ||\mathbf{x}_{ij}||$, where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. Hereafter, we call particles connected with other particles as connected particles. We correct velocities of connected particles only when their interparticle distance $||\mathbf{x}_{ij}||$ is larger than their initial interparticle distance r_{ij} (fluid expansion), namely $r_{ij} < ||\mathbf{x}_{ij}||$, because fluid compression, which generally occurs when $||\mathbf{x}_{ij}|| < r_{ij}$, is resolved to preserve fluid volumes with our fluid solver. To compute $\Delta\mathbf{v}$ for those connected particles such that $r_{ij} < ||\mathbf{x}_{ij}||$, we define a distance function D_{ij} as

$$D_{ij} = \max(||\mathbf{x}_{ij}|| - r_{ij}, 0).$$

Then, we compute $\Delta\mathbf{v}_i$ with a velocity correction coefficient c_i $(0 \leq l_i \leq c_i)$, where l_i is the lower limit of c_i:

$$\Delta\mathbf{v}_i = -\frac{1}{\Delta t}\sum_j^{S_i^f} \frac{c_i + c_j}{2}\frac{m_j}{m_i + m_j} D_{ij}\frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||}, \tag{6}$$

where S_i^f denotes the number of connected fluid particles to particle i. To derive Eq. (6), we adopted an idea of position correction used in PBD [4,20], where position correction vector is defined as

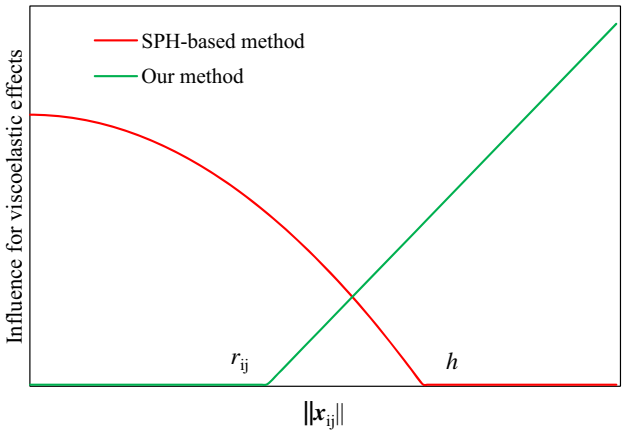

Fig. 3 Comparison of the influence for viscoelastic effects between an SPH-based method and our method

$$\Delta\mathbf{x}_i = -\sum_j^{S_i^f} \frac{m_j}{m_i + m_j}(||\mathbf{x}_{ij}|| - r_{ij})\frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||}.$$

Our velocity correction vector is basically obtained by dividing $\Delta\mathbf{x}_i$ by Δt. Using D_{ij}, the velocity correction vector becomes zero when $||\mathbf{x}_{ij}|| < r_{ij}$. We additionally introduce velocity correction coefficients c_i and c_j to control the stiffness of viscoelastic materials; lower (higher) c_i leads to softer (stiffer) materials (see Fig. 2).

Our velocity corrections enable large deformations that cannot be generated with SPH-based methods, e.g., [6,17], because the influence of the SPH-based methods for viscoelastic effects decreases as interparticle distances get larger and finally becomes zero owing to finite support kernels when $h < ||\mathbf{x}_{ij}||$ while the influence of our method increases when $r_{ij} < ||\mathbf{x}_{ij}||$ (see Fig. 3).

3.2.2 Coefficient relaxation

When large stress is applied to viscoelastic materials, they cannot completely return to their original shapes (this is known as plastic deformation). To simulate this behavior, we weaken the effect of c_i if $l_i < c_i$, taking into account the fact that connected particles do not restore their positional relation when their connection is extended larger than a certain distance. Specifically, we update c_i by

$$c_i^{t+1} = \max\left(c_i^t - \Delta t \sum_j^{S_i^f} \frac{w_i + w_j}{2}\sigma_{ij}, l_i\right), \quad (7)$$

$$\sigma_{ij} = \begin{cases} 1 & \text{if } \frac{\gamma_i+\gamma_j}{2} < \frac{D_{ij}}{h} \\ 0 & \text{otherwise} \end{cases},$$

where $w_i (0 \leq w_i)$ controls the weakening speed of c_i, and $\gamma_i (0 \leq \gamma_i)$ is a yield criterion. σ_{ij} works so that the weakening happens only when normalized extension D_{ij}/h is larger than $\frac{\gamma_i+\gamma_j}{2}$.

3.2.3 Connection control

We adaptively generate and delete particle connections depending on the positional relations of particles to allow viscoelastic fluids to merge with others and split into several lumps. For merging, we generate a new connection between two particles with α_i $(0 < \alpha_i)$ if $||\mathbf{x}_{ij}||/h < \frac{\alpha_i+\alpha_j}{2}$, and if there is no connection between them. For splitting, on the other hand, we remove particle connections with $\beta_i (0 < \beta_i)$ if $\frac{\beta_i+\beta_j}{2} < ||\mathbf{x}_{ij}||/h$. We show our connection control algorithm in Algorithm 1 for clarity. Merging and splitting of viscoelastic materials are demonstrated in Fig. 4.

Algorithm 1 Connection control algorithm

```
for all fluid particle i do
  for all neighboring particle j do
    if ||x_ij||/h < (α_i+α_j)/2 then
      if there is no connection then
        generate a new connection between i and j
for all connected particle j do
  if (β_i+β_j)/2 < ||x_ij||/h then
    delete the connection between i and j
```

3.2.4 Boundary-handling for solids

We simulate interactions of viscoelastic fluids and solids by incorporating solid boundary particles into our algorithm. Connections between fluid and solid particles are updated following Algorithm 1. We compute $\Delta\mathbf{v}_i$ by extending Eq. (6) with solid particles:

$$\Delta\mathbf{v}_i = -\frac{1}{\Delta t}\sum_j^{S_i^f} \frac{c_i + c_j}{2}\frac{m_j}{m_i + m_j} D_{ij}\frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||} - \frac{1}{\Delta t}\sum_k^{S_i^s} \frac{c_i + c_k}{2}\frac{m_k}{m_i + m_k} D_{ik}\frac{\mathbf{x}_{ik}}{||\mathbf{x}_{ik}||}, \quad (8)$$

where k denotes a neighboring solid particle and S_i^s the number of connected solid particles to particle i. Since force-based rigid body simulators are often adopted, we compute viscoelastic force $\mathbf{F}_i^v$ to exert the antisymmetric effect of fluid particles to solid particle i, preserving their momentum:

$$\mathbf{F}_i^v = -\frac{m_i}{\Delta t^2}\sum_j^{S_i^f} \frac{c_i + c_j}{2}\frac{m_j}{m_i + m_j} D_{ij}\frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||}. \quad (9)$$

Computing velocity correction vector $\Delta\mathbf{v}_i$ for one-way solid-to-fluid coupling is straightforward. Assuming $m_k = \infty$ in Eq. (8) for a solid particle k, which is fixed or moves along a determined path, we obtain

$$\Delta\mathbf{v}_i = -\frac{1}{\Delta t}\sum_j^{S_i^f} \frac{c_i + c_j}{2}\frac{m_j}{m_i + m_j} D_{ij}\frac{\mathbf{x}_{ij}}{||\mathbf{x}_{ij}||} - \frac{1}{\Delta t}\sum_k^{S_i^s} \frac{c_i + c_k}{2} D_{ik}\frac{\mathbf{x}_{ik}}{||\mathbf{x}_{ik}||}. \quad (10)$$

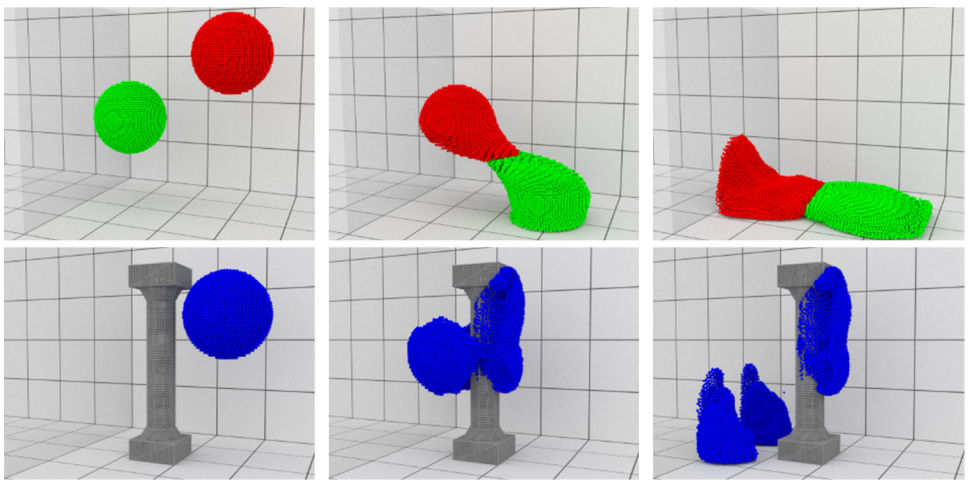

Fig. 4 (*Top*) Two merging viscoelastic balls. (*Bottom*) A viscoelastic ball thrown toward a column, splitting into three lumps

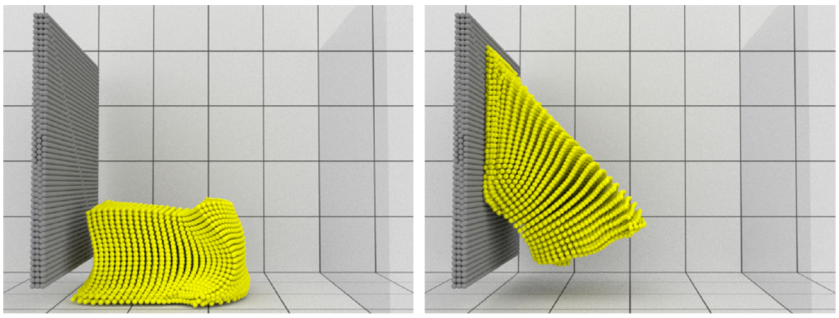

Fig. 5 A cube thrown toward a wall without (*left*)/with (*right*) fluid-solid coupling

Extending Eq. (7) with solid particles, we weaken the effect of c_i as

$$c_i^{t+1} = \max\left(c_i^t - \Delta t\left(\sum_j^{S_i^f} w_{ij}\sigma_{ij} + \sum_k^{S_i^s} w_{ik}\sigma_{ik}\right), l_i\right), \tag{11}$$

where $w_{ij} = \frac{w_i + w_j}{2}$.

Figure 5 illustrates the effects of the fluid-solid coupling with an example of a cube thrown toward a wall.

3.3 Simulation algorithm and implementation

We give an outline of our procedures in Algorithm 2. In our implementation, we use IISPH [12] as an incompressible fluid solver and set a convergence criterion as 0.01 %. For fluid-solid coupling of pressure and viscosity, we employ the method of Akinci et al. [1]. We use finite support kernels as proposed by Müller et al. [19]. To accelerate neighbor search steps, we used a variant of the z-index sort algorithm presented in [11]. There are six adjustable parameters c, l, w, γ, α, and β that we introduced to cover various phenomena of viscoelastic fluids, and we basically use values similar to $c = 0.001, l = 0.0001, w = 0.001, \gamma = 1.01, \alpha = 1.0$, and $\beta = 2.0$, adjusting them according to scenarios.

Algorithm 2 Simulation algorithm

```
for all particle i do
    find neighboring particles
for all particle i do
    update particle connections following Algorithm 1
    if particle i is a fluid particle then
        obtain v_i^* with Eqs. (5), (8), and (10)
        relax c_i with Eq. (11)
compute particle pressure p using IISPH
for all fluid particle i do
    compute F_i^p with Eq. (2)
    update solids with Eq. (9)
for all fluid particle i do
    integrate v_i with Eq. (3)
    integrate x_i with Eq. (4)
```

Table 1 Simulation conditions and performance

Figure #	N^f/N^s	t^{total} (s)
2 (left)	24.5k/38.6k	3.02
2 (middle)	24.5k/38.6k	3.83
2 (right)	24.5k/38.6k	4.06
4 (top)	45.2k/60.3k	9.57
4 (bottom)	22.6k/67.4k	5.36
5 (left)	13.8k/27.0k	2.40
5 (right)	13.8k/27.0k	2.43
6 (top)	7.1k/21.8k	0.39
6 (bottom)	7.1k/21.8k	0.64
7 (middle)	8.2k/19.2k	5.75
7 (right)	8.2k/19.2k	2.47
9 (middle)	14.0k/27.5k	3.35
9 (right)	14.0k/27.5k	3.13
11	Up to 195.1k/168.1k	125.2
12	33.4k/65.7k	6.82
13	73.5k/108.8k	29.01

N^f and N^s denote the number of fluid and solid particles, respectively. t^{total} is total simulation time per frame

4 Results

We implemented our method in C^{++} and parallelized it using OpenMP 2.0. All the simulations were performed on a PC with a 4-core Intel Core i7 3.40 GHz CPU and RAM 16.0 GB. We used a physical-based renderer Mitsuba to render all the scenarios. Our velocity corrections generally occupy only 1.3 % of whole simulation time. Simulation conditions and performance are listed in Table 1.

4.1 Large deformation

We compare results generated with our method and the SPH-based method using a non-linear corotational Maxwell model proposed by Mao and Yang [17] to show that our hybrid method can handle large deformations of a viscoelas-

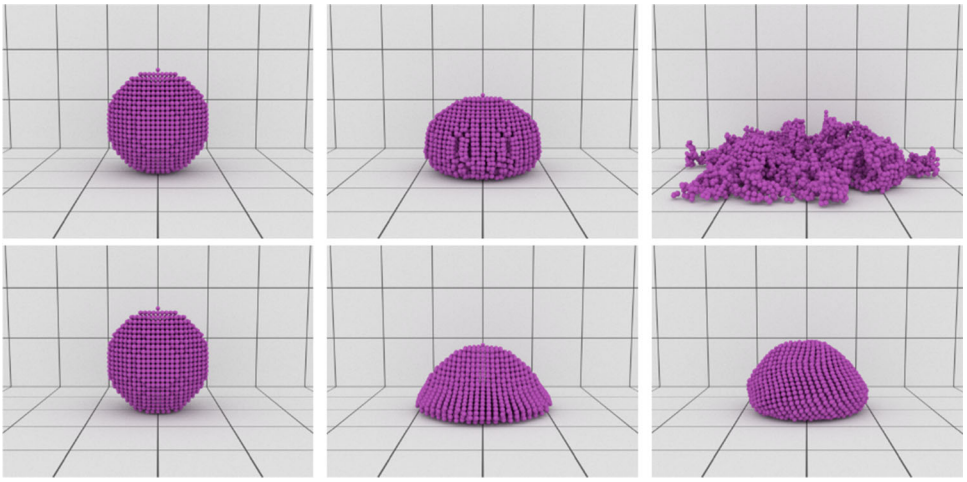

Fig. 6 Comparison for large deformation. (*Top*) the method of Mao and Yang [17]. (*Bottom*) our method

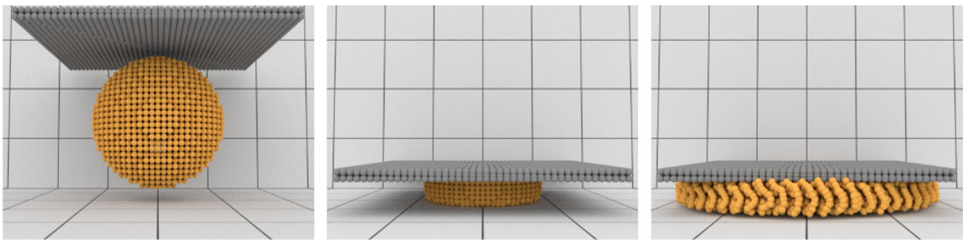

Fig. 7 Comparison for volume preservation. (*Left*) initial state. (*Middle*) the method of Takamatsu and Kanai [28]. (*Right*) our method

tic material, which cannot be simulated with the SPH-based method [17]. Though the method of Mao and Yang [17] resampled particles to solve an issue of particle disorder, we did not do that in our experiment to show that our method can generate large deformations without the need of special treatments.

Figure 6 illustrates a viscoelastic sphere dropped onto the ground. The viscoelastic ball simulated with the method of Mao and Yang [17] exhibits slight elastic deformations. After the degree of the deformations exceeds a certain threshold, the ball fails to restore their original shape and breaks into many lumps which consist of fairly close particles because particles close to others exert strong forces to preserve their positional relations (see Fig. 3). The method of Mao and Yang [17] needs to use strong viscoelastic forces to make the viscoelastic ball exhibit slight elastic deformations at the expense of numerical stability. On the other hand, our method can handle large deformations, restoring its original shape without resampling particles nor introducing numerical instability.

4.2 Volume preservation

We compare our method and the existing hybrid methods [7,28], using a scenario of a viscoelastic ball compressed by a moving board along a vertical path.

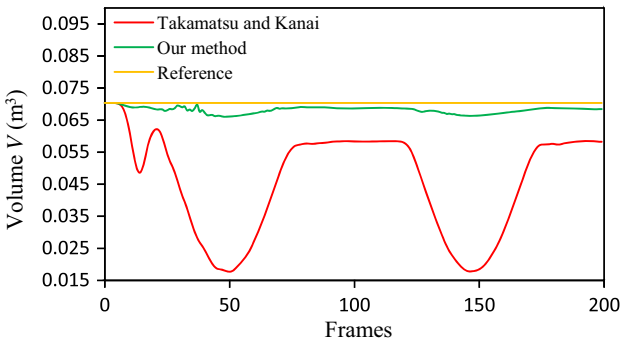

Fig. 8 Profiles of the volume V occupied by a ball for Fig. 7

Comparison to [28] First, we illustrate the difference between our method and the method of Takamatsu and Kanai [28], who combined SPH and SM. Since the basic SPH [19] adopted by Takamatsu and Kanai [28] as their underlying fluid solver cannot preserve fluid volumes, we use IISPH [12] instead of the basic SPH [19] and consider the method of [28] as a hybrid of IISPH and SM. In this comparison, we use 100 pressure iterations in IISPH for both methods because the method of [28] is likely to fail to preserve fluid volumes even if more pressure iterations are applied.

Figure 7 shows the comparison, where the left is the initial state, and the middle (right) is the result of the method [28] (our method), and Fig. 8 illustrates the volume V (which is

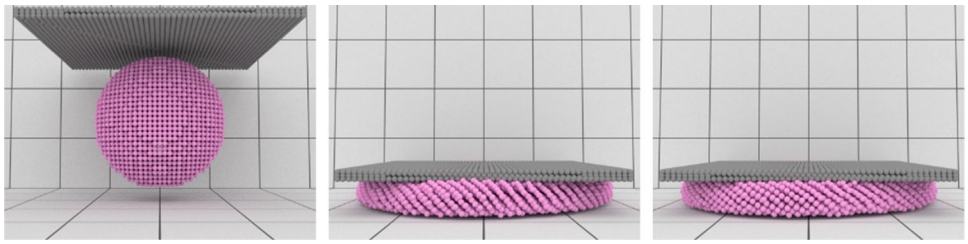

Fig. 9 Comparison for volume preservation. (*Left*) initial state. (*Middle*) the method of Clavet et al. [7] (*Right*) our method

estimated by summing up the volume of all fluid particles as $V = \sum_i V_i = \sum_i m_i/\rho_i$) occupied by the ball for the method of Takamatsu and Kanai [28], our method, and reference (initial volume). The volume of the ball on the middle decreases and increases depending on the moving board. By contrast, the ball on the right preserves its volume, which is fairly close to the reference. Additionally, when being compressed, the ball on the right exhibits horizontally spreading motions, which cannot be generated with the method of [28].

Moreover, our method is advantageous over the method of [28] in terms of performance (see Table 1), and there are two main factors; first, the method of Takamatsu and Kanai [28] needs more neighboring particles to stabilize simulations than our method, and processing these particles is an additional cost. Second, their method [28] uses Singular Value Decomposition, which is time consuming, to obtain rotational matrices in SM while our velocity corrections need simple computations only.

Comparison to [7] Second, we compare our method and the method of Clavet et al. [7], who combined SPH with spring systems using a prediction-relaxation scheme. Similar to the above previous method, since Clavet et al. [7] used the basic SPH [19], we use IISPH as their fluid solver for the same reason.

Figure 9 demonstrates the comparison, where the left is the initial state, and the middle (right) is the result of the method [7] (our method). Since errors introduced by position changes during the iteration are relatively small and it is difficult to clarify visual or volume differences between the previous method [7] and our method, we compare these methods in terms of the number of iterations required to satisfy the convergence criterion of 0.01 %, as shown in Fig. 10, where profiles of iteration numbers for Clavet et al. [7] n^{spr} and our method n^{pos}, and their difference $n^{\mathrm{spr}} - n^{\mathrm{pos}}$ are illustrated. Although our method and the method of Clavet et al. [7] can generate similar results, the method of Clavet et al. [7] is likely to require more iterations to enforce fluid incompressibility than our method. This is noticeable especially when the ball is compressed and significantly deformed by the board because large deformations cause strong attraction forces and introduce larger positional errors (see "Difference" in Fig. 10). The extra iterations of the method of

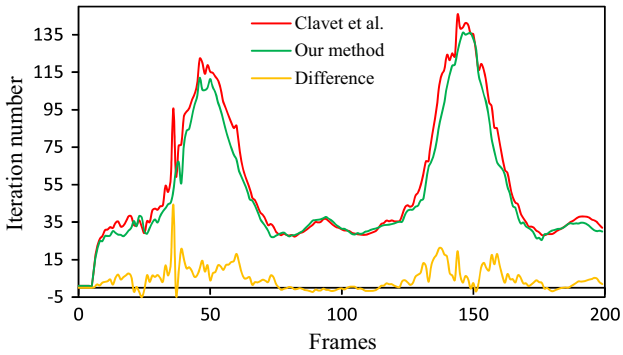

Fig. 10 Profiles of iteration number required to satisfy the convergence criterion of 0.01 % for Fig. 9

Clavet et al. [7] lead to slightly higher computational cost than our method (see Table 1).

4.3 Interactions

Figure 11 illustrates several spheres with different viscoelasticity values successively thrown toward a static solid dragon. Similar to Figs. 4 (top) and 5 (right), our method can easily handle merging of fluids and adhesion of fluids to a solid by updating pairwise interparticle connections.

Figure 12 demonstrates two-way interactions of a viscoelastic material and a solid. The solid cube is gradually pulled toward a wall by the viscoelastic material that connects to both of the solid cube and the wall. This effect cannot be simulated only with the previously proposed boundary-handling scheme for pressure and viscosity [1].

4.4 Barus effect

Figure 13 demonstrates Barus effect of a viscoelastic ball. Barus effect (also known as die swell, exclude swell, and Merrington effect) is a phenomenon that a stream of viscoelastic fluid swells wider than the diameter of an opening when the stream goes through the opening because of viscoelastic forces which restore fluid's original shape after the deformations at the opening. Although our velocity correcting method is based on approximated dynamics of viscoelastic fluid, we can plausibly generate Barus effect.

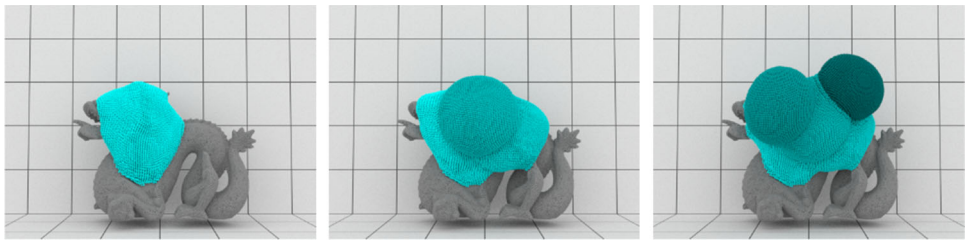

Fig. 11 Spheres with different viscoelasticity values successively thrown toward a solid dragon. Particles are color coded based on their velocity correction coefficients c (low coefficient *cyan* and high coefficient *dark cyan*)

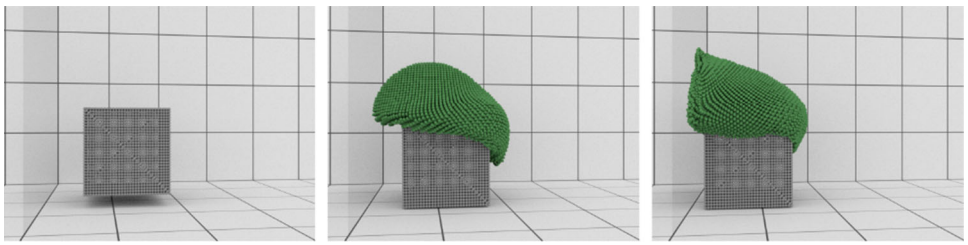

Fig. 12 A viscoelastic ball thrown toward a solid cube

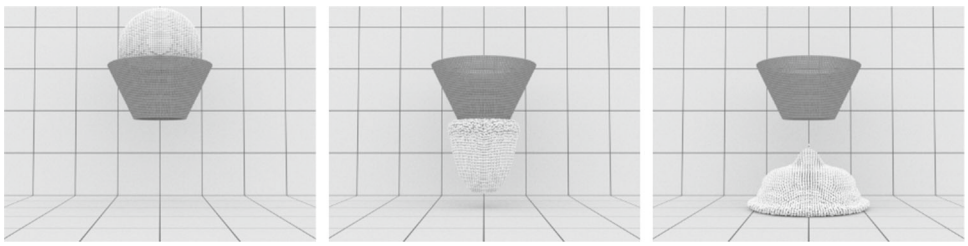

Fig. 13 Barus effect of a viscoelastic material

5 Discussions and limitations

Our method combines PBD with SPH to take advantage of geometric methods, such as numerical stability and capability of handling large deformations. Additionally, unlike SPH-based methods, our method has another benefit that time steps are not generally restricted by viscoelasticity values, namely velocity correction coefficient c in our method; and thus our method can use larger time steps and be faster than SPH-based methods. In contrast to positive aspects, however, there are a few negative issues on the adaptation of PBD. First, our method exploits PBD to approximate viscoelastic behaviors instead of using physical-based models. Because of this approximation, our simulation results are less accurate than SPH-based methods which use a physical-based model for viscoelastic effects. Second, we introduce several parameters which lack physical meanings into our method to cover various effects of viscoelastic fluids. These parameters need to be adjusted depending on each scenario through experiments to generate desirable behaviors of materials. Third, since results of our position-based velocity correction can be affected by time steps similar to other geometric methods, our method would produce different behavior of viscoelastic fluids depending on time steps.

When materials are driven to extreme situations (e.g., a ball forcibly and significantly deformed by a heavy object), particle penetrations can be observed because of interparticle connections which exert larger velocity changes for farther connected particles to enable large deformations. Although this can be addressed by adjusting the influence of velocity corrections, such an adjustment could make viscoelastic materials fail to restore their original shapes from highly deformed shapes.

Most viscoelastic materials undergo phase and property changes, e.g., turning into a stiff solid, as time passes. To simulate stiff materials, strengthening the influence of velocity corrections would cause instability in simulations. Thus, investigating how to simulate soft and hard materials with phase and property changes in a unified framework is our important future work.

Acknowledgments This work has been partly supported by JST CREST. We would like to thank anonymous reviewers for their valuable suggestions and comments.

References

1. Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B., Teschner, M.: Versatile rigid-fluid coupling for incompressible SPH. ACM Trans. Graph. **31**(4), 62:1–62:8 (2012)
2. Becker, M., Ihmsen, M., Teschner, M.: Corotated SPH for deformable solids. In: Proceedings of the Fifth Eurographics Conference on Natural Phenomena, pp. 27–34 (2009)

3. Becker, M., Teschner, M.: Weakly compressible SPH for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 209–217 (2007)
4. Bender, J., Müller, M., Otaduy, M.A., Teschner, M.: Position-based methods for the simulation of solid objects in computer graphics. In: EUROGRAPHICS 2013 State of the Art Reports, pp. 1–22 (2013)
5. Bodin, K., Lacoursiere, C., Servin, M.: Constraint fluids. IEEE Trans. Vis. Comput. Graph. **18**(3), 516–526 (2012)
6. Chang, Y., Bao, K., Liu, Y., Zhu, J., Wu, E.: A particle-based method for viscoelastic fluids animation. In: Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, pp. 111–117 (2009)
7. Clavet, S., Beaudoin, P., Poulin, P.: Particle-based viscoelastic fluid simulation. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 219–228 (2005)
8. Cummins, S.J., Rudman, M.: An SPH projection method. J. Comput. Phys. **152**(2), 584–607 (1999)
9. Gerszewski, D., Bhattacharya, H., Bargteil, A.W.: A point-based method for animating elastoplastic solids. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 133–138 (2009)
10. He, X., Liu, N., Li, S., Wang, H., Wang, G.: Local poisson SPH for viscous incompressible fluids. Comput. Graph. Forum **31**(6), 1948–1958 (2012)
11. Ihmsen, M., Akinci, N., Becker, M., Teschner, M.: A parallel SPH implementation on multi-core CPUs. Comput. Graph. Forum **30**(1), 99–112 (2011)
12. Ihmsen, M., Cornelis, J., Solenthaler, B., Horvath, C., Teschner, M.: Implicit incompressible SPH. IEEE Trans. Vis. Comput. Graph. **20**(3), 426–435 (2014)
13. Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A., Teschner, M.: SPH fluids in computer graphics. In: EUROGRAPHICS 2014 State of the Art Reports, pp. 21–42 (2014)
14. Jones, B., Ward, S., Jallepalli, A., Perenia, J., Bargteil, A.W.: Deformation embedding for point-based elastoplastic simulation. ACM Trans. Graph. **33**(2), 21:1–21:9 (2014)
15. Macklin, M., Müller, M.: Position based fluids. ACM Trans. Graph. **32**(4), 104:1–104:5 (2013)
16. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Trans. Graph. **33**(4), 104 (2014)
17. Mao, H., Yang, Y.H.: Particle-based non-Newtonian fluid animation with heating effects. Tech. rep. (2006)
18. Miller, G., Pearce, A.: Globular dynamics: a connected particle system for animating viscous fluids. Comput. Graph. **13**(3), 305–309 (1989)
19. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 154–159 (2003)
20. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. J. Vis. Commun. Image Rep. **18**(2), 109–118 (2007)
21. Müller, M., Heidelberger, B., Teschner, M., Gross, M.: Meshless deformations based on shape matching. ACM Trans. Graph. **24**(3), 471–478 (2005)
22. Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., Alexa, M.: Point based animation of elastic, plastic and melting objects. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 141–151 (2004)
23. Solenthaler, B., Pajarola, R.: Predictive-corrective incompressible SPH. ACM Trans. Graph. **28**(3), 40:1–40:6 (2009)
24. Solenthaler, B., Schläfli, J., Pajarola, R.: A unified particle model for fluid-solid interactions. Comput. Animat. Virtual Worlds **18**(1), 69–82 (2007)
25. Steele, K., Cline, D., Egbert, P.K., Dinerstein, J.: Modeling and rendering viscous liquids. Comput. Animat. Virtual Worlds **15**(3–4), 183–192 (2004)
26. Takahashi, T., Fujishiro, I., Nishita, T.: A velocity correcting method for volume preserving viscoelastic fluids. In: Proceedings of the Computer Graphics International 2014 (2014)
27. Takahashi, T., Nishita, T., Fujishiro, I.: Fast simulation of viscous fluids with elasticity and thermal conductivity using position-based dynamics. Comput. Graph. **43**, 21–30 (2014)
28. Takamatsu, K., Kanai, T.: A fast and practical method for animating particle-based viscoelastic fluids. Int. J. Virtual Real. **10**, 29–35 (2011)
29. Tamura, N., Nakaguchi, T., Tsumura, N., Miyake, Y.: Spring-bead animation of viscoelastic materials. IEEE Comput. Graph. Appl. **27**(6), 87–93 (2007)
30. Terzopoulos, D., Platt, J., Fleischer, K.: Heating and melting deformable models. J. Vis. Comput. Animat. **2**, 68–73 (1991)
31. Zhou, Y., Lun, Z., Kalogerakis, E., Wang, R.: Implicit integration for particle-based simulation of elasto-plastic solids. Comput. Graph. Forum **32**(7), 215–223 (2013)