A Fast and Practical Method for Animating Particle-Based Viscoelastic Fluids

Kenji Takamatsu and Takashi Kanai

The University of Tokyo, Graduate School of Arts and Sciences

***Abstract*—This paper proposes a practical technique for fast animation of materials such as viscoelastic fluids. A fast animation of such materials is desperately desirable especially for real-time applications such as games. We compute the behavior of viscoelastic fluids approximately instead of the exact simulation by combining two well-established approaches, Smoothed-Particle Hydrodynamics and Shape Matching. This enables fast and stable computations. A combination is done by a simple linear interpolation of velocities. A variety of materials between a fluid and an elastic solid can be represented by changing only a parameter of linear interpolation. We also propose how to bring our approximate method closer to the actual motions of viscoelastic fluids including merging or splitting of objects. We demonstrate a high-speed performance of our method with presenting several interesting results.**

***Index Terms*—Computer Animation, Viscoelastic Fluids, Particle-Based Simulation, Smoothed-Particle Hydrodynamics, Shape Matching.**

I. INTRODUCTION

In this paper we describe a practical technique for fast animation of materials such as viscoelastic fluids. Viscoelastic fluids are the materials which have both physical properties of fluids and elastic solids. By weak forces they keep their original shape like an elastic solid, and by strong forces they deform and change their shape like a fluid. There are a huge variety of materials which represent this type of behavior, e.g., clay, chewing gum, toothpaste, shaving cream, gelatin, etc. Animations of these materials have recently been successfully used in special effects for computer graphics applications. Especially for games, fast animation of such materials is desperately desirable.

According to the theory of continuum mechanics [7], the difference between a perfect fluid and an elastic solid is whether an elastic force, which an object reinstates itself to its original shape, is included or not. To simulate the behavior of viscoelastic fluids in the literature, the general approaches are to introduce elastic forces into the Navier-Stokes equation. Indeed, previous approaches in CG community solve such extended governing equations by using an Eulerian grid-based fluid simulation [11] or a particle-based Lagrangian fluid simulation [10]. These straightforward approaches are, however, hard to be processed in real-time. As far as we know, there is no approach to simulate viscoelastic fluids fast enough to be used in real-time applications.

We propose a fast and stable method to compute the behavior of viscoelastic fluids approximately instead of the exact simulation. A key idea here is to combine two well-established approaches for fast and stable computations of object motions. For computing fluid motions, a Smoothed-Particle Hydrodynamics (SPH) method [14] can be recently used. It is a particle-based Lagrangian method and then each particle can be moved freely. On the other hand, Shape Matching (SM) methods [17, 21] approximately represent motions of elastic solids in real-time. They are originally designed for solids, i.e., the connectivity of elements (particles) does not change during deformations. However, those methods are by nature extendable in the case of changing the connectivity of particles. A combination is done by a simple linear interpolation of current velocities in order to keep the high-speed performance and the robustness of an individual method. Consequently, a variety of materials between a fluid and an elastic solid can be changed by only a parameter of linear interpolation. We also discuss how to bring our approximate method closer to the actual motions of viscoelastic fluids including merging or splitting of objects.

II. RELATED WORK

2.1 Fluid simulation

Fluid simulation became widely known in computer graphics by a method of Foster and Metaxas [9]. Their method solves Navier-Stokes equation, the governing equation of fluids, by discretizing using an Eulerian grid. Stam [24] simplified an advective term by using the semi-Lagrangian method to improve an Eulerian grid-based method with robustly taking a large time step. On the other hand, Lagrangian particle methods such as Smoothed-Particle Hydrodynamics (SPH) [14] were well studied recently. SPH was first introduced in astrophysics and Müller et al. [15] successfully used in computer graphics at

Manuscript Received on 25 December 2010.
E-mail: kanait@acm.org

interactive rates. Adams et al. [1] used an adaptive sampling method to improve computational performance. Performances were further improved by using GPUs [12]. SPH can also be used for representing other types of materials between fluids and solids. For example, Solenthaler et al. [23] additionally introduced a temperature term to represent melting and solidification of objects.

2.2 Elastic solid animation

Elastic solid animation was introduced by Terzopoulos et al. [26] using a finite difference method. Several other methods were also studied such as a mass-spring method [2], a FEM method [16], a particle-based method [18]. Those methods are, however, time-consuming due to exact and robust solutions of elasticity equation. On the other hand, a Shape Matching (SM) method originally proposed by Müller et al. [17] is a geometry-based approach and imitates an elastic deformation. The main advantage is its fast and unconditionally stable computation; there is no need to solve the equation of motion. Later, its computational performance was further improved by using an adaptive sampling [25] or by using a lattice shape [21]. As an example to imitate physical motions, Rungjiratananon et al. [22] recently extended SM to simulate human's hairstyles. Becker et al.[4] used both SPH and SM to represent elastic motions. This is most relevant research to ours in the sense that SPH and SM can be efficiently combined. The main difference is that they integrate the computation of rotation matrices in SM into SPH, while our method simply interpolate velocities so as to keep the high-speed performance and the robustness of an individual method.

2.3 Viscoelastic Fluids / Viscoplastic solids simulation

Goktekin et al. [11] realized a viscoelastic fluid simulation by taking into account an elastic term to an Eulerian grid-based fluid simulation [8]. Bargteil et al. [3] achieved a robust viscoplastic solid simulation by using a FEM method and remeshing. Several methods based on SPH were also studied for considering elasticity, plasticity, and viscosity. Clavet et al. [6] added springs between pairs of neighboring particles in SPH. Paiva et al. [19] modified the traditional N-S equation and employed generalized Newtonian liquid model to simulate viscoplastic fluids. Solenthaler et al. [23] introduced a unified particle model for the simulation of liquids and deformable solids as well as rigid objects. This is the most relevant research to ours. Chang et al. [5] introduced more general elastic stress term to the N-S equation and changed the viscosity and elastic stress coefficients according to the temperature variation. Gerszewski et al. [10] applied arbitrary constitutive models to compute elastic forces in viscoplastic solids by using deformation gradients. All these methods are, however, hard to be used in real-time applications. This is mainly because each time step has to be set to an extremely small value to robustly handle numerical simulations.

III. ANIMATION FRAMEWORK

In this section, we describe our animation framework to compute the behavior of viscoelastic fluids. As described in Section 1, a key idea is to combine two well-established approaches for fast and stable computations. It should be noted that our approach does not solve a combined N-S equation with an additional elastic term like as most of previously-published approaches. In our approach two fast and stable approaches, SPH for fluid simulation and SM for elastic solid deformation, are processed independently in each simulation step and two velocities are linearly interpolated. A new position is then computed by integrating an interpolated velocity. Note that we do not consider the plastic deformations, since we use SM to represent elastic motions approximately.

3.1 Combination of SPH and SM

We briefly introduce two approaches, SPH and SM, at first. We then describe how to combine these two approaches.

SPH formulations. SPH is an interpolation method with each particle carrying field quantities. A force for each particle p_i $(\mathrm{i} = 1 \ldots n^p)$ is computed based on the physical properties of neighboring particle p_j weighted by kernel functions. According to the N-S equation, forces for each particle are the pressure force $\boldsymbol{f}_i^{press}$, the viscosity force $\boldsymbol{f}_i^{visco}$, and the external force $\boldsymbol{f}_i^{ext}$ including gravity force and collision response forces as follows:

$$\boldsymbol{f}_i^{press} = -\sum_j \frac{m_j}{\rho_j} p_j \nabla W_{\sigma^s}\left(\left|\boldsymbol{x}_j - \boldsymbol{x}_i\right|\right) \tag{1}$$

$$\boldsymbol{f}_i^{visco} = \mu \sum_j \frac{m_j}{\rho_j} \boldsymbol{v}_j \nabla^2 W_{\sigma^s}\left(\left|\boldsymbol{x}_j - \boldsymbol{x}_i\right|\right) \tag{2}$$

where m_i, $\rho_i = \sum_j m_j W_{\sigma^s}\left(\left|\boldsymbol{x}_j - \boldsymbol{x}_i\right|\right)$, $\boldsymbol{v}_i$ are mass, density, and velocity vector respectively. Also, $p_i = k(\rho_i - \rho_0)$, ρ_0, k, μ denote pressure, initial density, pressure coefficient, and viscosity coefficient respectively. Note that neighboring particles p_j have to be updated for each simulation step. A kernel function $W_{\sigma^s}(r)$ is defined as follows:

$$W_{\sigma^s}(r) = \frac{315}{64\pi(\sigma^s)^9}((\sigma^s)^2 - r^2)^3 \tag{3}$$

Based on three forces described above, an acceleration vector $\boldsymbol{a}_i^{SPH}$ is calculated as follows:

$$\boldsymbol{a}_i^{SPH} = \frac{\left(\boldsymbol{f}_i^{press} + \boldsymbol{f}_i^{visco} + \boldsymbol{f}_i^{ext}\right)}{\rho_i} \tag{4}$$

A position and a velocity vector are updated from such an acceleration vector by using the standard Euler method as follows:

$$\boldsymbol{v}_i^{SPH} = \boldsymbol{v}_i^t + \Delta t^{SPH} \boldsymbol{a}_i^{SPH} \tag{5}$$

$$\boldsymbol{x}_i^{SPH} = \boldsymbol{x}_i^t + \Delta t^{SPH}\boldsymbol{v}_i^{SPH} \quad (6)$$

SM formulations. SM imitates an elastic solid deformation. Figure 1 illustrates the original SM scheme. The reference shape $\boldsymbol{x}^{ref}$ is rigidly transformed to its goal position $\boldsymbol{g}$ by using a rotation matrix R and a transformation vector $\boldsymbol{t}$. We compute $\boldsymbol{t}$ as a barycenter position, and R by the polar decomposition of a linear transformation matrix from $\boldsymbol{x}^{ref}$ to $\boldsymbol{x}$. A position $\boldsymbol{x}$ of each particle is then pulled towards its goal position $\boldsymbol{g}$.

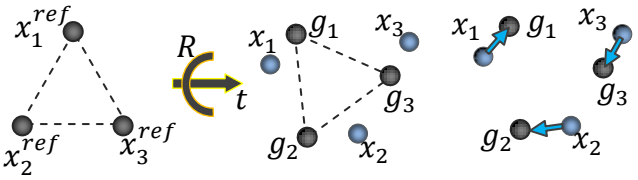


Fig. 1. Computation of R and t in SM.

Here we slightly extend a method proposed by Rivers and James [21] to fit our animation framework. The original scheme utilizes a lattice structure to improve computational performance. In contrast, our extension adapts the case that neighboring particles are arbitrary located.

For a particle p_i ($i = 1 \ldots n^p$), neighboring particles $p_j \in N_i$ within a support sphere of radius σ^s are collected. A goal position g_i is then defined as the average of rigidly transformed positions from neighboring particles,

$$g_i = \frac{1}{|N_i|}\sum_j \left(R_j\left(x_i^{ref} - x_j^{ref}\right) + t_j\right) \quad (7)$$

where R_j and t_j are a rotation matrix and a transformation vector of rigid motion in each neighbor particle p_j.

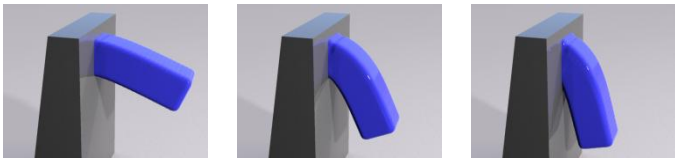


Fig. 2. Comparison to the motions in SM with different settings of σ^s. The length of the bar is 40.0. From left to right: $\sigma^s = 8.0$, $\sigma^s = 5.0$, $\sigma^s = 3.0$.

Fig. 2 demonstrates the results for different support radii σ^s. A larger support size makes an object stiffer due to the effect of more particles. The computation time also increases much more for a larger support size.

A position and a velocity vector are updated from a goal position as follows:

$$\boldsymbol{v}_i^{SM} = \boldsymbol{v}_i^t + \frac{(\boldsymbol{g}_i - \boldsymbol{x}_i^t)}{\Delta t^{SM}} + \Delta t^{SM}\frac{\boldsymbol{f}_i^{ext}}{m_i} \quad (8)$$

$$\boldsymbol{x}_i^{SM} = \boldsymbol{x}_i^t + \Delta t^{SM}\boldsymbol{v}_i^{SM} \quad (9)$$

Combination by the interpolation of velocities. In our combination method, velocities of both SPH and SM are firstly updated independently by Equation (5) and (8). Such two velocities are linearly interpolated by using only a parameter α ($0 \le \alpha \le 1$). A new position is then computed by using an Euler integration scheme as follows:

$$\boldsymbol{v}_i^{t+\Delta t} = \alpha\boldsymbol{v}_i^{SPH} + (1-\alpha)\boldsymbol{v}_i^{SM} \quad (10)$$

$$\boldsymbol{x}_i^{t+\Delta t} = \boldsymbol{x}_i^t + \Delta t\boldsymbol{v}_i^{t+\Delta t} \quad (11)$$

In addition to the interpolation of velocities described above, the use of acceleration vectors or positions can be considered for the combination. However, there is a possibility that it is computationally unstable due to the division by a small Δt.

Our method can represent various types of materials with different physical properties by changing a parameter α. Figure 3 compares the shapes of cubes with different α when they are fallen on the floor. As shown in this figure, a cube deforms like an elastic solid with $\alpha = 0.0$, and a cube flows like a fluid with $\alpha = 1.0$. Also, a viscoelastic behavior can be presented when α is set to an intermediate value between 0 and 1. An elastic property is greatly appeared as like a jelly with $\alpha = 0.3$, and a fluid property is stronger as like a toothpaste with $\alpha = 0.7$ in Figure 3.

Adjusting the movement of particles. In SPH, a time interval Δt^{SPH} is dynamically changed to keep the simulation stable. Δt^{SPH} is controlled so as not to move larger than a support radius of a particle in each simulation step, i.e., Δt^{SPH} is set in order to satisfy the following inequation;

$$\Delta t^{SPH} \cdot \max\left(\left|\boldsymbol{v}_i^{SPH}\right|\right) < \sigma^s \quad (12)$$

where $\max\left(\left|\boldsymbol{v}_i^{SPH}\right|\right)$ denotes a maximum value of the magnitude of velocities for all particles. If a velocity is large, a time interval is set to a small value and then the movement distance of a particle in each step becomes small.

On the other hand, Δt^{SM} has little effect on the movement of particles in SM. In Equation (8) a velocity becomes large for a small Δt^{SM}. However, in Equation (9) a position is updated by adding a velocity multiplied with Δt^{SM}, then the effect of Δt^{SM} gets balanced out. Consequently, the effect of a fluid over the elasticity is relatively changed with different settings of Δt^{SPH}.

To resolve this issue, the movements of particles in SM are adjusted by a time interval Δt^{SPH}. That is, Equation (8), a formula for computing the velocity, is re-written as follows:

$$\boldsymbol{v}_i^{SM} = \boldsymbol{v}_i^t + \frac{\Delta t^{SPH}}{\Delta t_0^{SPH}}\left(\frac{(\boldsymbol{g}_i - \boldsymbol{x}_i^t)}{\Delta t^{SM}} + \frac{\Delta t^{SM}\boldsymbol{f}_i^{ext}}{m_i}\right) \quad (13)$$

The second term on the right of Equation (8) is scaled to follow the dynamic change of Δt^{SPH} over its initial value Δt_0^{SPH}. Therefore, a position and a velocity in SM are automatically controlled in a balanced manner. The adjustment of a time interval occurs when the density of particles becomes high, e.g. a collision against other objects. In this case, velocities of particles become large due to the high pressure forces. In our experiments, we set $\Delta t_0^{SPH} = \Delta t^{SM} = 0.15$ and a 50 percents smaller Δt^{SPH} in maximum than Δt_0^{SPH} is observed during the simulation.

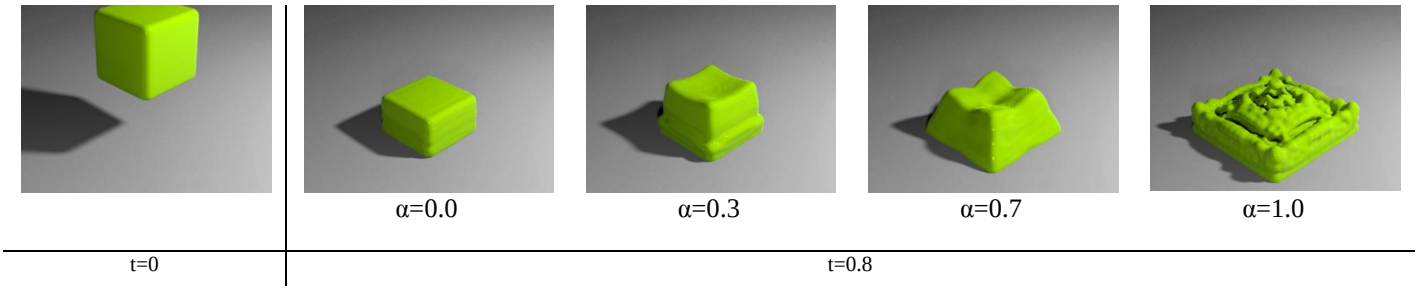


Fig. 3. Comparison of the shapes of fallen cubes with various setting of α.

3.2 Splitting and Merging

Splitting or merging frequently occurs in the viscoelastic materials. A viscoelastic object in general is split into two small objects when external forces stronger than internal elastic forces are applied to a part of its body. Two objects are merged when external forces of objects collided with each other exceed over their internal forces.

Since in SPH each particle moves freely and the arrangement of particles is not fixed, splitting and merging naturally occur. However, in the original SM, a reference shape is used to keep its original shape as an elastic solid. The arrangement of particles in such a reference shape is fixed during the simulation. Therefore, splitting or merging never occurs due to the fixed reference shape. We apply here the following two extensions to establish splitting and merging with SM.

Update reference shape based on material properties. We update a reference shape during the simulation in contrast to the original SM. When a reference shape is updated, neighboring particles in each particle are possibly changed. Splitting or merging can occur according to the relationship between neighbor particles. Note that the computational cost of such update is subtle since the neighboring particles are already constructed in SPH and can be reused.

Several factors are considered to check whether the reference shape is updated or not. Firstly, the change of the object shape is one of key factors. Here we consider external forces adding to an object. This is because that a topological change of a viscoelastic fluid is thought to be caused by suffering external forces. We then check whether a reference shape is updated or not by the magnitude of external forces. Let $|\boldsymbol{f}^{ext}|$ be an average of the magnitude of external forces for all particles. A reference shape is updated if $|\boldsymbol{f}^{ext}| > f_u$, where f_u denotes a threshold. It should be noted that f_u is an important parameter to check the update of the reference shape, e.g., setting a larger f_u tends to be harder to update the reference shape.

Another key factor to check the update of the reference shape is the physical property of an object. In the case of the elastic solid, a reference shape does not want to be updated despite huge external forces. Also, a reference shape wants to be updated for each step in case of a fluid. To satisfy both demands, we relate a parameter α to f_u. For a small α an object is close to an elastic solid, and then f_u should be set to a large value. On the contrary, it is desirable for a fluid to set f_u to a small value for a large α.

We then define a monotonically decreasing function as shown in Figure 4 to compute f_u according to α as follows:

$$f_u = \gamma_u(1-\alpha) \tag{14}$$

where γ_u denotes a value of f_u for $\alpha = 0$. This function is especially useful in the animation which α is varied continuously.

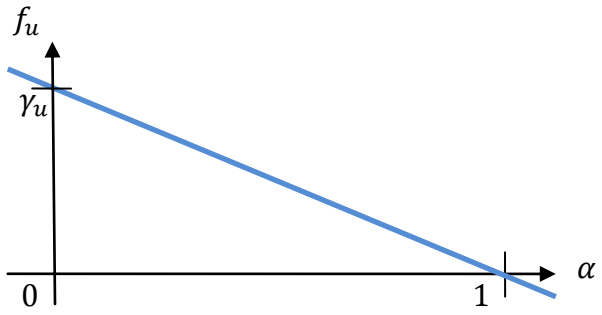


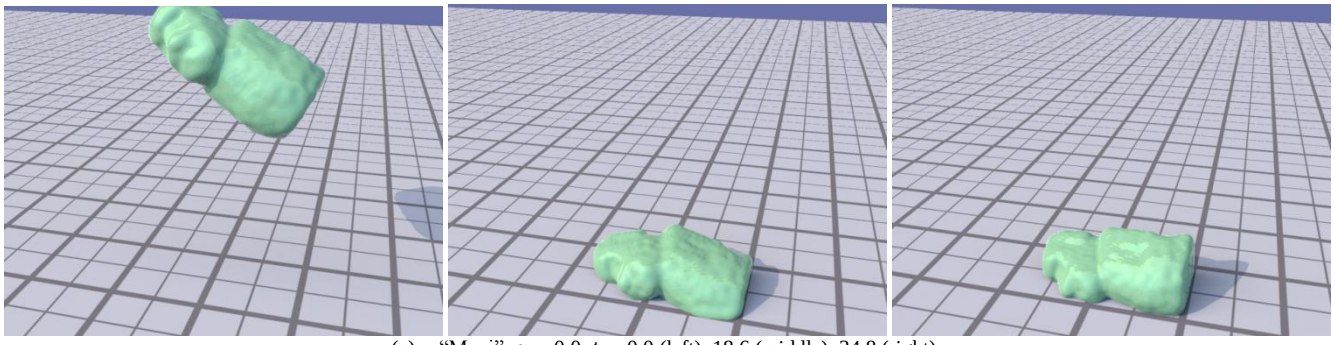
Fig. 4. A $f_u - \alpha$ function to compute f_u according to α.

Setting the number of simulation steps for updating reference shape. If the update of the reference shape is applied in every simulation step, the reference shape is deformed like a fluid. A viscoelastic motion cannot then be realized. So, it is better to have a certain interval to check the update. Here we introduce a parameter m_u and check the update if the number of simulation steps reaches m_u. We empirically set m_u 2-3 times larger than fps in our simulator to work our check well at reasonable computation time.

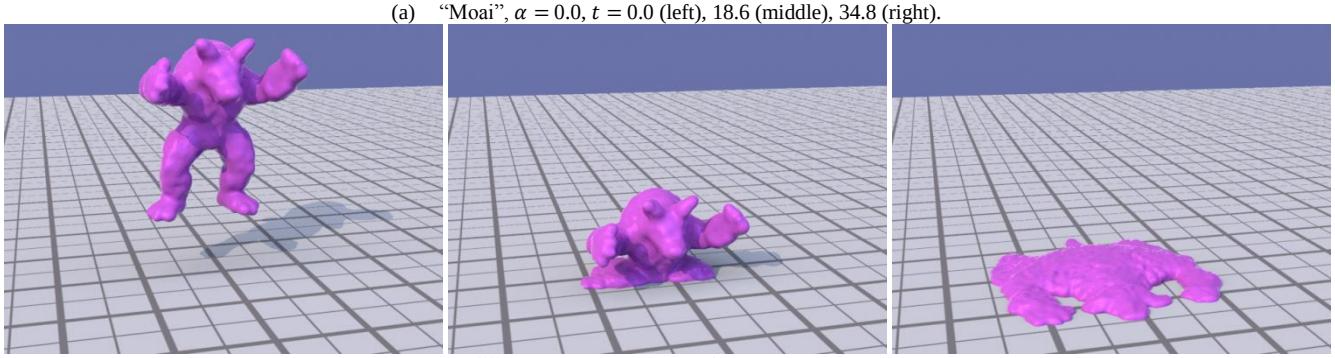
3.3 Algorithm

We now describe our whole algorithm below. c denotes the number of simulation steps to be used for checking the update of the reference shape.

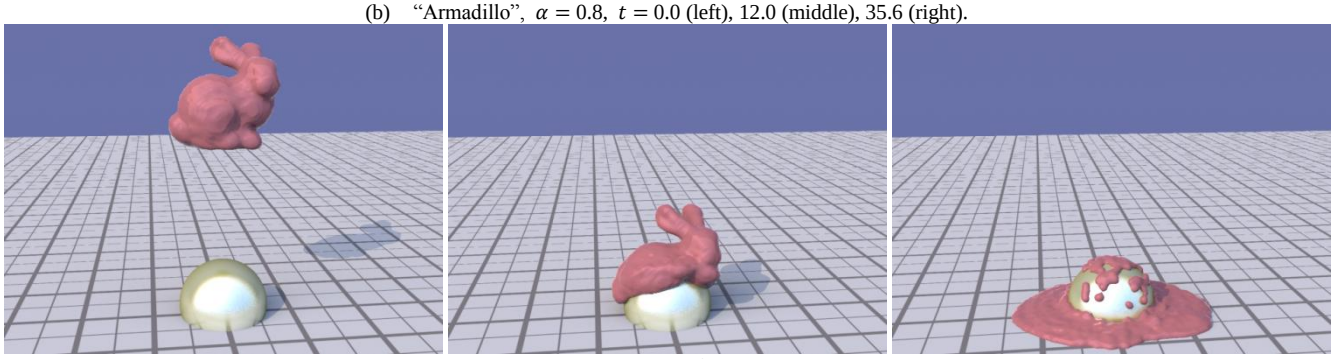
$c \leftarrow 0$;
loop
  if $c \bmod m_u = 0$ **then**
    if $|\boldsymbol{f}^{ext}| > f_u$ **then**
      $\boldsymbol{x}^{ref} \leftarrow \boldsymbol{x}^t$; {Update of the ref. shape}
    end if
  end if
  Adjust Δt^{SPH}; {Eq. (12)}



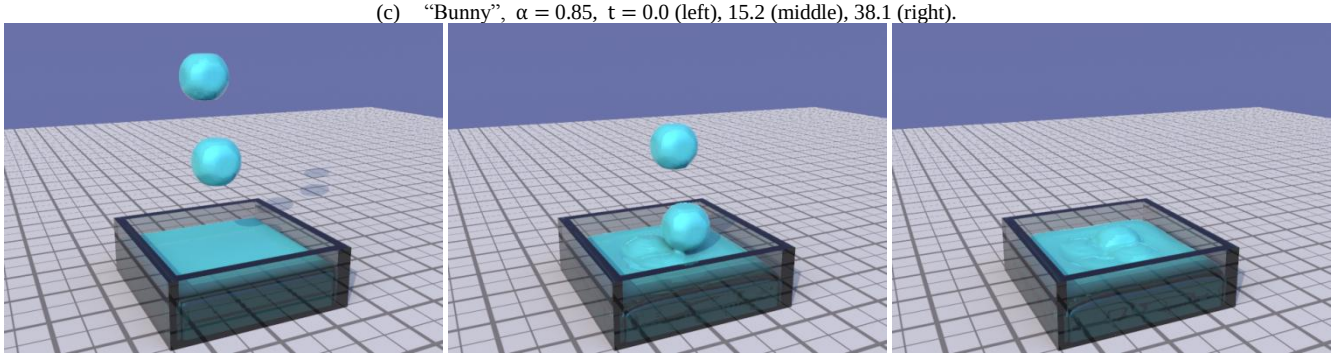
(a) "Moai", $\alpha = 0.0$, $t = 0.0$ (left), 18.6 (middle), 34.8 (right).



(b) "Armadillo", $\alpha = 0.8$, $t = 0.0$ (left), 12.0 (middle), 35.6 (right).



(c) "Bunny", $\alpha = 0.85$, $t = 0.0$ (left), 15.2 (middle), 38.1 (right).



(d) "Four balls in a pool", $\alpha = 0.9$, $t = 0.0$ (left), 26.5 (middle), 43.25 (right).

Fig. 5. Experimental results of our method with different settings of α.

Compute $\boldsymbol{v}^{SPH}$; {Eq. (5)}
Compute $\boldsymbol{v}^{SM}$; {Eq. (8)}
Compute $\boldsymbol{v}^{t+\Delta t}$, $\boldsymbol{x}^{t+\Delta t}$; {Eq. (10), (11)}
(Option) Rendering by $\boldsymbol{x}^{t+\Delta t}$;
$\boldsymbol{v}^{t} \leftarrow \boldsymbol{v}^{t+\Delta t}$;
$\boldsymbol{x}^{t} \leftarrow \boldsymbol{x}^{t+\Delta t}$;
$c \leftarrow c + 1$;
end loop

3.4 Rendering

Just after positions are updated, we render the surface of the current particles. Although a lot of methods for the rendering of SPH particles have been recently proposed, we adopt a simple method. We first create an implicit distance field on a regular grid covering particles, and then extract an iso-surface by using Marching Cubes algorithm [13].

TABLE 1: STATISTICAL RESULTS OF OUR EXPERIMENTS. n^p: THE NUMBER OF PARTICLES. $|\tilde{N}|$: THE AVERAGE NUMBER OF NEIGHBORING PARTICLES. V-SIZE: THE SIZE OF VOXELIZATION FOR SURFACE EXTRACTION. FPS (1): FPS WITHOUT SURFACE EXTRACTION AND RENDERING. FPS (2): FPS WITH SURFACE EXTRACTION AND RENDERING.

	Fig. 2 Bar			Fig. 3 Cube		Fig. 5(a) Moai		Fig. 5(b) Armadillo		Fig. 5(c) Bunny		Fig. 5(d) Balls
n^p	3,600	3,600	3,600	10,648	3,375	3,764	2,015	4,157	2,680	4,185	1,328	8,264
$\|\tilde{N}\|$	755.3	256.5	64.3	80.7	23.6	85.3	43.1	27.1	16.6	35.1	26.7	28.3
α	0.0	0.0	0.0	0.7	0.7	0.0	0.0	0.8	0.8	0.85	0.85	0.9
γ_u	8.0	8.0	8.0	12.0	12.0	6.0	6.0	6.0	6.0	6.0	6.0	10.0
V-size	96	96	96	128	128	160	64	160	64	160	32	160
x	96	96	96	128	128	160	64	160	64	160	32	160
y	128	128	128	128	128	192	96	192	96	192	48	224
z												
FPS (1)	1.8	4.7	14.2	4.1	25.1	8.7	26.6	21.9	42.2	17.4	64.6	11.3
FPS (2)	1.3	2.4	3.8	0.35	1.2	0.56	5.4	0.68	6.3	0.42	36.8	0.20

To achieve fast rendering, we first extract a part of particles which are on the surface, and a distance field is then created from those particles. In SPH, a particle on the surface tends to have less neighboring particles than an inner particle, and then its density is lower. Therefore we consider as a particle on the surface if its density ρ_i is less than a threshold ρ^s.

IV. RESULTS AND DISCUSSION

We discuss our results in this section. All our experiments were performed using a notebook PC with Intel Core 2 Duo P8700 2.53GHz CPU and nVIDIA GeForce GT 130M GPU. Table 1 presents the statistical results of our experiments.

Our input is a set of 3D solid points. To create uniformly-sampled points from polygonal meshes, we used 3D Delaunay triangulations in CGAL [20]. Note that we used only vertices of the output tetrahedra as our input. Resulting images are created by using Sunflow [27], an open source global illumination renderer.

Fig. 5(a) shows the "Moai" model (3,764 points) with setting $\alpha = 0.0$. As can be seen from this figure, its motions are like an elastic solid and keep its original shape even after being bounced on the floor. It is to be noted that a support size is set to a large value to establish stiffer motions, and the average number of neighboring particles therefore becomes large. Also, a SM in our method is obviously slower than the original method in [21], because we adapt our method to the case that neighboring particles are arbitrary located.

Fig. 5(b) shows the "Armadillo" model (4,157 points) with setting $\alpha = 0.8$. Its motions are like an elastic solid but a fluid property is also included. As shown in this figure, an object is collided and is spread on the floor; however, its shape is not perfectly collapsed.

Fig. 5(c) shows the "Bunny" model (4,185 points) with setting $\alpha = 0.85$. Its motions are like a fluid with some elasticity. This experiment presents an example of splitting and merging; we can see that an object is once collided with a hemisphere and is split into several parts. They are finally merged on the floor.

Fig. 5(d) shows four balls dropped in the pool with setting $\alpha = 0.9$. Its motions are also like a fluid with a little bit elasticity. It can be seen that a ball is dropped and is merged into the pool water.

Concerning about the computational performance of our method, most time-consuming part is caused by computing positions in SM, especially when the number of neighboring particles is large. Our method is still competitive because it keeps more than 5 times faster compared to other recently proposed methods for viscoelastic motions [19, 5, 10]. The method by Paiva et al. [19] is the fastest among three approaches. The computation of "Pressing Cube" composed of approximately 6K particles is at 1.81 FPS on a Centrino 1.86GHz CPU as shown in [19]. In contrast our method establishes 14 FPS by a similar experiment.

V. CONCLUSION AND FUTURE WORK

We have proposed a practical technique for fast animation of viscoelastic fluids based on combining a fluid simulation by SPH and an elastic deformation by SM. Setting a parameter α realizes various types of materials between a fluid and an elastic solid. Splitting and merging can be also presented by controlling the update of the reference shape in SM. Our method achieves high computational performance with the ease of changing various types of materials.

In future work, we would like to implement our method on GPUs or multi-core CPUs. We think that it dramatically improves the computational performance even on a stand-alone PC. Another future work is that we would like to extend our simulator to deal with several different settings of materials at a time.

REFERENCES

[1] B. Adams, M. Pauly, R. Keiser and L. J. Guibas. Adaptively Sampled Particle Fluids. *ACM Transaction on Graphics*, 26(3), pp.48:1-48:7, 2007.

[2] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. *Proc. ACM SIGGRAPH '98*, pp.43-54, 1998.

[3] A. W. Bargteil, C. Wojtan, J. K. Hodgins and G. Turk. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Transaction on Graphics*, 26(3), pp.16:1-16:8, 2007.
[4] M. Becker and M. Ihmsen and M. Teschner. Corotated SPH for deformable solids. *Proc. Eurographics Workshop on Natural Phenomena*, pp. 27-34, 2009.
[5] Y. Chang, K. Bao, Y. Liu, J. Zhu and E. Wu. A Particle-Based Method for Viscoelastic Fluids Animation. *Proc. 16th ACM Symposium on Virtual Reality Software and Technology*, pp. 111-117, 2009.
[6] S. Clavet, P. Beaudoin and P. Poulin. Particle-based Viscoelastic Fluid Simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 219-228, 2005.
[7] E. H. Dill. Continuum Mechanics: Elasticity, Plasticity, Viscoelasticity. CRC Press, 2006.
[8] D. Enright, S. Marschner and R. Fedkiw. Animation and Rendering of Complex Water Surfaces. *ACM Transaction on Graphics*, 21(3), pp. 736-744, 2002.
[9] N. Foster and D. Metaxas. Controlling Fluid Animation. *Proc. Computer Graphics International*, pp. 178-188, 1997.
[10] D. Gerszewski, H. Bhattacharya and A. W. Bargteil. A Point-Based Method for Animating Elastoplastic Solids. *Proc. ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, pp. 133-138, 2009.
[11] T. G. Goktekin, A. W. Bargteil and J. F. O'Brien. A Method for Animating Viscoelastic Fluids. *ACM Transaction on Graphics*, 23(3), pp. 463-468, 2004.
[12] P. Goswami, P. Schlegel, B. Solenthaler and R. Pajarola. Interactive SPH Simulation and Rendering on the GPU. *Proc. ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, pp. 55-64, 2010.
[13] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Proc. ACM SIGGRAPH '87*, pp. 163-169, 1987.
[14] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Reports on Progress in Physics*, 68(8), pp. 1703-1759, 2005.
[15] M. Müller, D. Charypar and M. Gross. Particle-Based Fluid Simulation for Interactive Applications. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 154-159, 2003.
[16] M. Müller, J. Dorsey, L. McMillan, R. Jagnow and B. Cutler. Stable Real-Time Deformations. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 49-54, 2002.
[17] M. Müller, B. Heidelberger, M. Teschner and M. Gross. Meshless Deformations Based on Shape Matching. *ACM Transaction on Graphics*, 24(3), pp.471-478, 2005.
[18] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross and M. Alexa. Point Based Animation of Elastic, Plastic and Melting Objects. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 141-151, 2004.
[19] A. Paiva, F. Petronetto, T. Lewiner and G. Tavares. Particle-Based Viscoplastic Fluid/Solid Simulation. *Computer-Aided Design*, 41(4), pp. 306-314, 2009.
[20] S. Pion and M. Teillaud. 3D Triangulations. *CGAL User and Reference Manual*. CGAL Editorial Board, 3.7 edition, 2010.
[21] A. R. Rivers and D. L. James. FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM Transaction on Graphics*, 26(3), pp.82:1-82:6, 2007.
[22] W. Rungjiratananon, Y. Kanamori and T. Nishita. Chain Shape Matching for Simulating Complex Hairstyles. *Computer Graphics Forum*, 29(8), pp. 2438-2446, 2010.
[23] B. Solenthaler, J. Schläfli and R. Pajarola. A Unified Particle Model for Fluid-Solid Interactions. *Computer Animation and Virtual Worlds*, 18(1), pp.69-82, 2007.
[24] J. Stam. Stable Fluids. *Proc. ACM SIGGRAPH '99*, pp. 121-128, 1999.
[25] D. Steinemann, M. A. Otaduy and M. Gross. Fast Adaptive Shape Matching Deformations. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 87-94, 2008.
[26] D. Terzopoulos, J. Platt, A. Barr and K. Fleischer. Elastically Deformable Models. *Proc. ACM SIGGRAPH '87*, pp. 205-214, 1987.
[27] SunFlow: The Open Source Render Engine. http://sunflow.sourceforge.net/.

Kenji Takamatsu is a master course student in the Graduate School of Arts and Sciences, the University of Tokyo, Japan. His research interests include physics-based modeling for CG animations.

Takashi Kanai is an associate professor in the Graduate School of Arts and Sciences, the University of Tokyo, Japan. His research interests include geometric modeling and its application to computer graphics. He received his doctor degree of engineering from the University of Tokyo in 1998. He is a member of ACM, IEEE CS, JSPE (Japan Society for Precision Engineering), IPSJ (Information Processing Society of Japan).