Derivative Particles for Simulating Detailed Movements of Fluids

Oh-young Song, Doyub Kim, and Hyeong-Seok Ko

Abstract— **We present a new fluid simulation technique that significantly reduces the non-physical dissipation of velocity. The proposed method is based on an apt use of particles and derivative information. We note that a major source of numerical dissipation in the conventional Navier–Stokes equations solver lies in the advection step. Hence, starting with the conventional grid-based simulator, when the details of fluid movements need to be simulated, we replace the advection part with a particle simulator. When swapping between the grid-based and particle-based simulators, the physical quantities such as the level set and velocity must be converted. For this purpose, we develop a novel dissipation-suppressing conversion procedure that utilizes the derivative information stored in the particles as well as in the grid points. For the fluid regions where such details are not needed, the advection is simulated using an octree-based constrained interpolation profile (CIP) solver, which we develop in this work. Through several experiments, we show that the proposed technique can reproduce the detailed movements of high-Reynolds-number fluids, such as droplets/bubbles, thin water sheets, and whirlpools. The increased accuracy in the advection, which forms the basis of the proposed technique, can also be used to produce better results in larger scale fluid simulations.**

Index Terms— **physically based modeling, multiphase fluid, high-Reynolds-number fluid, water**

I. Introduction

When water interacts violently with solids, air, or water itself, it takes on a variety of structures, including droplets/bubbles, thin water sheets, and whirlpools, as shown in Figure 1. Such features can appear when fluids undergo motions characterized by high Reynolds numbers, where the Reynolds number represents the relative magnitude of the inertia compared to the viscosity of the fluid. Fast-moving water is a typical high-Reynolds-number fluid. This paper explores the physically-based simulation of such phenomena.

Oh-young Song is with Sejong University. Doyub Kim and Hyeong-Seok Ko are with Seoul National University. Email: oysong@sejong.ac.kr, {kim,ko}@graphics.snu.ac.kr

Assuming that the Navier–Stokes equations correctly model the physical movements of fluids, a plausible simulation should be able to reproduce the high-Reynolds-number behaviors of water.[1] To date, however, these phenomena have not been satisfactory reproduced. This paper identifies the factors underlying this failure, and proposes a method that allows realistic simulation of high-Reynolds-number liquid motions.

Implausible simulation of high-Reynolds-number fluids is related to *numerical dissipation*. Specifically, discretized simulation of the Navier–Stokes equations inevitably calls for evaluating physical quantities at non-grid points. In most methods, such values are calculated by interpolating the values of the physical quantities at the grid points. However, the error introduced by this approximation acts like nonphysical viscosity or numerical dissipation. This dissipation can be reduced by using a finer grid; however, increasing the grid resolution may increase the computation time and memory requirements to impractical levels. Over the last few years, several elegant techniques have been proposed to address this issue. Bringing in particles into the Eulerian scheme can help capture the inertial movement of fluids, and increases the effective resolution of the simulation. Enright et al. [1] introduced the particle level set method, which increases the accuracy of surface tracking by spreading particles around the interface. Losasso et al. [2] proposed an octree-based multi-level fluid solver that allows finer-resolution simulations in more interesting regions such as water surfaces.

Unfortunately, the above techniques cannot produce high-Reynolds-number liquids with sufficient detail and realism. The simulated fluid appears more viscous than in real physics; the fluids often look thick/sticky, and the movements of small-scale

[1]Although the present work focuses on water, it can apply to any fluid.

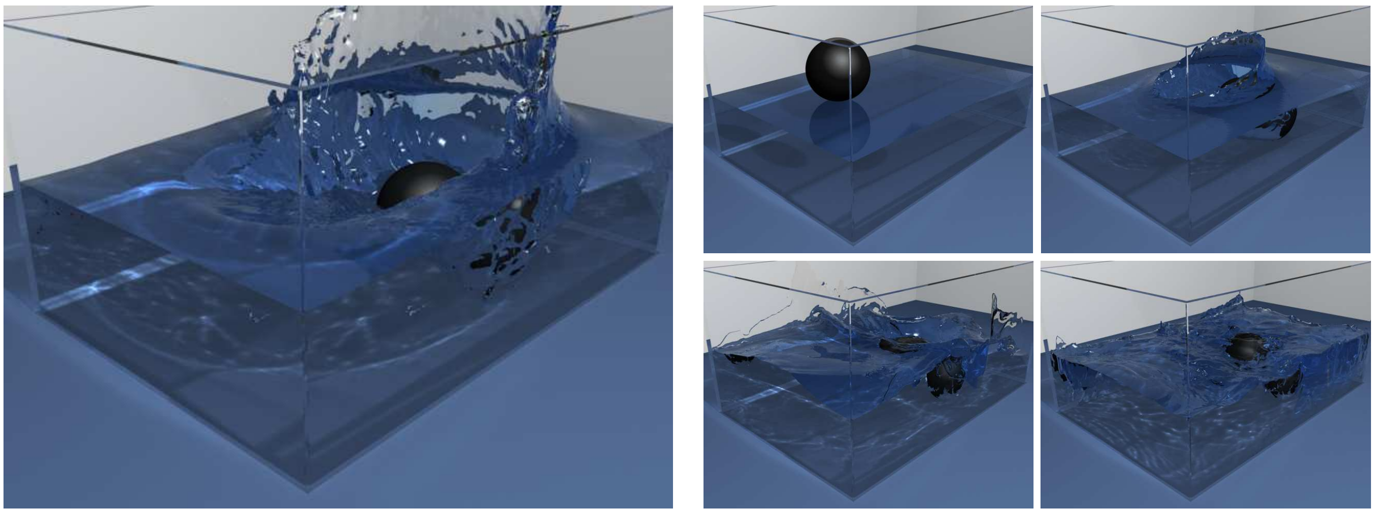


Fig. 1. Detailed movements of water when a ball makes an impact into the water. The sequence was produced by the method proposed in this paper. $192 \times 128 \times 128$ effective resolution was used.

features that are typically observed in complex flows do not appear. We find that the reason the above dissipation-suppressing techniques produce this sort of artifacts is because they cannot effectively suppress the *velocity*-dissipation, even though they reduce the mass-dissipation remarkably.

In this paper we introduce a new concept called *derivative particles*, and develop a fluid simulator based on that concept. We take advantage of the non-dissipative nature of the Lagrangian scheme for the simulation of the advection part; for fluid regions where details need to be simulated, we solve the advection step using particles. Switching between the grid-based and particle-based simulators can introduce numerical dissipation. This paper develops a new conversion procedure that allows the reproduction of detailed fluid movements. One innovative aspect of this work is that, in addition to storing the physical quantities (velocity and level set value), the derivative particles also store the spatial derivatives of those quantities, which enables more accurate evaluation of the physical quantities at non-grid/non-particle positions. The use of particles in the present work differs from the particle level set method [1] in that the derivative particles carry fluid velocities as well as level set values.

Experiments show that the proposed simulator tracks interfaces accurately. More interestingly, the proposed method turns out to reduce nonphysical damping to a remarkable extent, allowing reproduction of the magnificent movements of small scale features that occur in real high-Reynolds-number fluids.

The remainder of the paper is organized as follows: Section II reviews previous work; Section III gives an overview of the simulator; Section IV presents the octree-based constrained interpolation profile (CIP) solver; Section V presents the derivative particle model; Section VI reports our experimental results; and Section VII concludes the paper.

II. Previous Work

Since Foster and Metaxas [3], [4] first introduced fluid animation techniques based on full 3D Navier–Stokes simulation, the approach has spread widely among the computer graphics community. Jos Stam [5] introduced a stable fluid solver known as *Stable Fluids*. The advection step of this solver was implemented using a semi-Lagrangian method [6], which remains stable even when large time steps are used. Since then, there have been active developments of fast fluid animation techniques in computer graphics, based on the semi-Lagrangian method. Rasmussen et al. [7] proposed a technique to produce 3D large-scale animations of gases using a 2D semi-Lagrangian solver, and Feldman et al. [8] proposed an explosion model that incorporates a particle-based combustion model into the Stable Fluids solver. Treuille et al. [9], [10] introduced an optimization technique that generates the fluid flows that meet specified keyframe constraints. In addition, Feldman et al. proposed the use of a hybrid grid that combines a staggered grid and an unstructured grid [11], and also proposed the use of a deformable grid [12], for which they had to modify the semi-Lagrangian method.

In order to simulate liquids, a model for tracking the liquid surfaces is needed in addition to a stable Navier–Stokes solver. For this purpose, Foster and Fediw [13] proposed a hybrid surface model in which massless particles are introduced to the level set field. This model motivated the development of the *particle level set method* [1], which can capture the dynamic movements of fluid surfaces with remarkable accuracy. Enright et al. [14] demonstrated that the particle level set method allows the use of large time steps in the semi-Lagrangian framework. The method has been employed to model the interactions between fluids and rigid bodies [15] and between fluids and deformable thin-shell objects [16], as well as to animate viscoelastic fluids [17], sand [18], multi-phase fluids [19], and water drops [20].

As an alternative to the grid-based approaches, purely particle-based methods have also been studied. Stam and Fiume [21] used *smoothed particle hydrodynamics* (SPH) to model gaseous phenomena. In SPH, the fluid is represented by a collection of particles, and the simulator calculates movements of them by accounting for each term of the Navier–Stokes equations. Müller et al. [22] used the SPH model for simulating liquids, and in [23], they use the technique for simulating multi-phase fluid interactions. Premože et al. [24] introduced the *moving particle semi-implicit* (MPS) method to the graphics community, a technique that shows better performance than SPH in simulating the incompressible movements of fluids. These purely particle-based approaches are suited for simulating unrestricted domains. However, surface reconstruction can be complicated in these approaches compared to the grid-based method.

A primary factor that impairs the visual realism (and also the physical accuracy) of simulated results is the numerical dissipation of the velocity field. To reduce the dissipation when simulating gaseous phenomena, Fedkiw et al. [25] used cubic interpolation. They also included an additional step referred to as *vorticity confinement* that amplifies the curl of the velocity field, producing realistic swirly components in smoke movements. More physically-based prevention of the vorticity dissipation was done [26], [27] by employing the the *vortex particle method*. This method works with the curl version of the Navier–Stokes equations and utilizes particles to solve the advection term, which results in effective preservation of the vorticities. In the case of liquids, however, viscosity plays a more prominent role; in particular, swirly movements are short-lived and less frequently observed. Thus, modeling the behavior of high-Reynolds-number liquids requires a velocity-dissipation suppression technique that works in more general situations. To reduce dissipation in liquid simulations, Kim et al. [28] proposed an advection technique using Back and Forth Error Compensation and Correction [29], which provides second-order accuracy in both space and time. Song et al. [30] proposed a technique based on the CIP advection method. Their technique solves the velocity advection with third-order accuracy. However, the numerical viscosity that is coming from the grid-based advection cannot be avoided by this approach. Zhu and Bridson [18] proposed a fluid simulation technique based on the FLuid-Implicit-Particle (FLIP) method. The FLIP method [31] solves the advection step with particles, but all other steps with the grid. Although the particle advection in this method is free from numerical dissipation, interpolation errors are introduced when the velocity values are transferred back and forth between the grid and particles.

III. Overview

In the development of a multi-phase fluid solver, we assume that both air and water are incompressible. The Navier–Stokes equations for incompressible fluids can be written as

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{\nabla p}{\rho} + \nabla \cdot (\mu \nabla \mathbf{u}) + \frac{\mathbf{f}}{\rho}, \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2)$$

where $\mathbf{u}$ is the velocity, p is the pressure, $\mathbf{f}$ is the external force, μ is the viscosity coefficient, and ρ is the fluid density. For accurate modeling of the discontinuity in the density and viscosity across the interface between the media, we employ the ghost fluid method with variable density [19], [32], [33]. In our solver, surface tracking is based on the level set method. The level set field ϕ is updated according to

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \quad (3)$$

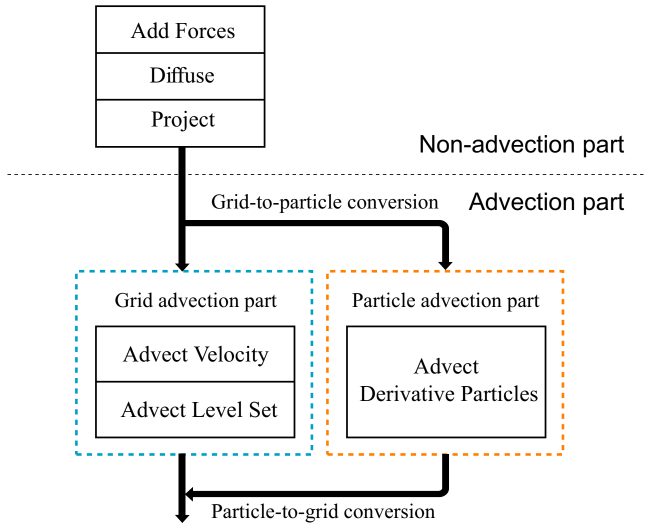


Fig. 2. Architecture of the simulator

with signed-distance condition

$$|\nabla\phi| = 1. \tag{4}$$

For the time-integration of the Navier–Stokes equations, we employ the fractional step method [5] which incrementally accounts for the effects of the terms in Equation (1) and the mass conservation condition in Equation (2). The technique we develop here focuses on the enhancement of the advection part. Hence we group the fractional steps into two parts, the non-advection part and the advection part, as shown in Figure 2. The time-integration of Equation (3) involves only the advection part.

The advection part consists of the grid advection part and the particle advection part. The grid advection part advects the velocity and level set fields calculated from the non-advection part according to the current velocity field. Our approach differs from pure Eulerian simulation in that our simulator includes the particle advection part, which is used to simulate regions where details need to be produced. For that purpose, we introduce a number of particles in the neighborhood of the air-water interface.[2]

In the grid advection part, Advect Velocity and Advect Level Set are Eulerian advection steps. Accurate handling of these steps forms the basis of successful simulation of high-Reynolds-number behaviors. For the Eulerian advection, we develop an octree-based CIP solver (Section IV) that reduces both velocity and mass dissipation to a remarkable level.

When the advection of a fluid region needs to be simulated using the particle advection part, the physical quantities (e.g., the level set and velocity) that are currently defined on the grid are transferred to the particles. After this transfer, the particles can be advected straightforwardly. The results of the particle advection are then transferred back to the grid. At this point in the procedure, the final velocity and level set values are stored on the grid points, regardless of whether the simulation proceeded via the grid advection part or particle advection part. We present the grid-to-particle and particle-to-grid velocity/level set conversion procedures in Section V. Use of the particle-based advection and development of the conversion procedures are essential for reproducing the details of fluid movements.

[2]The band was three-cells thick in each side of the interface; for 2D simulations we put 16 particles per cell, and for 3D simulations we put 32.

IV. Developing the Octree-based CIP Solver

This section describes the development of the grid advection part of the simulator by combining the CIP method with the octree data structure.

A. *Introduction to the CIP method*

In solving the Navier–Stokes equations with the fractional step method (Section III), the advection terms $\mathbf{u}\cdot\nabla\mathbf{u}$ and $\mathbf{u}\cdot\nabla\phi$ require special attention due to their hyperbolic nature. The semi-Lagrangian method provides a stable framework to simulate the above hyperbolic equations [5]. Unfortunately, this method suffers from severe numerical diffusion arising from the linear interpolation, which is used to determine the physical quantities at the back-tracked non-grid point from those at neighboring grid points.

Meanwhile, the CIP method is a third-order technique originally proposed by Yabe and Aoki [34], [35] and subsequently improved by Yabe et al. [36]. The key idea of this method is to advect not only the physical quantities but their derivatives as well. Here, a question would be how to advect the derivatives. An interesting observation of Yabe and Aoki [34] was that the equation for advecting derivatives can be obtained directly from the original hyperbolic equation

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0, \tag{5}$$

where ϕ is the physical quantity that is being advected. Differentiating Equation (5) with respect to the spatial variable *x*, we obtain

$$\frac{\partial \phi_x}{\partial t} + \mathbf{u} \cdot \nabla \phi_x = -\mathbf{u}_x \cdot \nabla \phi, \qquad (6)$$

which can be used to predict the evolution of ϕ_x over time. In solving Equation (6), once again we employ the fractional step method: first, we solve the non-advection part $\partial \phi_x / \partial t = -\mathbf{u}_x \cdot \nabla \phi$ using finite differencing; then we advect the result according to

$$\frac{\partial \phi_x}{\partial t} + \mathbf{u} \cdot \nabla \phi_x = 0. \qquad (7)$$

A detailed introduction to CIP advection can be found in [30]. The extension of the above approach to two- and three-dimensional cases is presented in [35]. Song et al. [30] found that the generalization given in [35] can lead to instabilities, and proposed a modification that fixes this problem.

In the present work, for solving the advection part, we employ the CIP method with 2nd-order Runge-Kutta time integration.

B. Combining CIP with Adaptive Octree Grids

The use of adaptive grids based on an octree data structure, which was introduced by Losasso et al. [2], allows simulations to be performed with inhomogeneous accuracy throughout the fluid. This approach is useful from a practical standpoint, since it allows more interesting regions such as surfaces to be simulated with higher accuracy by introducing small amounts of additional computation and memory. Here we adopt this octree data structure, and propose that the practical value of this adaptive approach can be further improved by combining it with the CIP method.

Losasso et al. [2] use linear interpolation for the semi-Lagrangian advection step. If, however, we replace the linear interpolation with CIP-interpolation, the advection will have third-order accuracy. As in Guendelman et al. [16], we sample the pressure value at the cell center, but sample all other quantities – velocity, level set, and their derivatives – at the nodes. When a cell is refined, the values at the new grid points are CIP-interpolated by referring to all the derivative values. Our octree coarsening/refining is identical to the procedure presented in [2].

We note that the CIP method fits very well with the octree data structure because, although it has third order spatial accuracy, it is defined over a single grid cell stencil rather than over multiple stencils. Due to this compactness, we can use the CIP method for simulating adaptive grids without any major modification. By contrast, extension of the method from first-order to third-order would be difficult for higher-order schemes that are defined over multiple stencils: as the simulation adaptively refines the grid, cells with different sizes will be produced, which makes the development of multi-stencil-higher-order schemes problematic.

We would also note a feature that we refer to as the *octree artifact*. Regional variations in the grid resolution of the octree data structure produce variations in the amount of dissipation. Regional variations in the mass dissipation may not be visually noticeable. For the velocity dissipation, however, the regional differences can be noticeable, especially for rapid fluid motions. In the simulation of breaking-dam, for example, water undergoes fast lateral movements. The fluid near the surface, where the resolution is high, experiences small amounts of numerical diffusion and thus makes swift movements. The fluid at the bottom, by contrast, moves in a manner characteristic of a more viscous fluid due to the low grid resolution. When these two types of motion appear together, the upper part of the water appears to be crawling over the lower part. Higher order schemes, including CIP, are not free from this kind of artifact. However, the artifact is less noticeable when the advection is simulated using the octree-based CIP solver proposed in the present work; we attribute this improvement to an overall reduction in numerical dissipation, especially in the low-resolution regions.

V. Derivative Particle Model

In the aspect of momentum/mass conservation, the particle-based Lagrangian scheme is more accurate than the grid-based Eulerian scheme. Hence, we apply the Lagrangian scheme to the regions that need to be simulated in greater detail. However, this approach has its own limitations in regard to surface tracking and pressure/viscosity calculation. Therefore we apply the approach only to the simulation of the advection part; all other parts of the simulation are based on the Eulerian scheme.

Switching between the Lagrangian and Eulerian schemes requires grid-to-particle and particle-to-grid conversion of the physical quantities (velocity and level set values). The procedures for performing these conversions should be carefully developed so that they do not introduce unnecessary numerical diffusion. In this section we present novel conversion procedures that do not impair the desirable characteristics of the Lagrangian advection, which is an essential component enabling the reproduction of the small scale features in high-Reynolds-number fluids. For simplicity, the description of this section is done for 2D.

A. Grid-to-Particle Velocity Conversion

At the end of the non-advection part shown in Figure 2, (1) the velocities that need to be advected are stored at the grid points, and (2) a number of particles are scattered over the grid. The particle advection part must find out the velocities of the particles so that their velocities and level set values make Lagrangian movements. Suppose that a particle $\mathbf{P}$ is in the cell defined by four grid points $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$, and $\mathbf{G}_4$. Let $\mathbf{u}_i^G = (u_i^G, v_i^G)$ be the grid velocities at $\mathbf{G}_i$ for $i \in \{1,2,3,4\}$. We are looking for the formula that can be used for the velocity $\mathbf{u}^P = (u^P, v^P)$ of $\mathbf{P}$.

One possible approach, which was used in the particle-in-cell method [37], would be to use the bi-linear interpolation $\mathbf{u}^P = \sum_{i=1}^4 w_i \mathbf{u}_i^G$, where w_i are the bi-linear weights determined based on the particle position $\mathbf{P}$. Unfortunately, this conversion introduces severe numerical diffusion. In the FLIP method [31], only the incremental parts of the grid velocities are transferred to the particle velocity. Namely, $\mathbf{u}^P = \mathbf{u}_-^P + \sum_{i=1}^4 w_i \Delta\mathbf{u}_i^G$, where $\mathbf{u}_-^P$ is the particle velocity just before the commencement of the non-advection part and $\Delta\mathbf{u}_i^G$ is the velocity change at $\mathbf{G}_i$ due to the non-advection part. This approach has been shown to reduce numerical diffusion conspicuously [18]. Hence, in the development of the grid-to-particle velocity conversion procedure, we use the FLIP approach, but with monotonic CIP interpolation [30] instead of linear interpolation. This not only gives $\mathbf{u}^P$ but also $(\frac{\partial}{\partial x}u^P, \frac{\partial}{\partial y}u^P)$ and $(\frac{\partial}{\partial x}v^P, \frac{\partial}{\partial y}v^P)$.

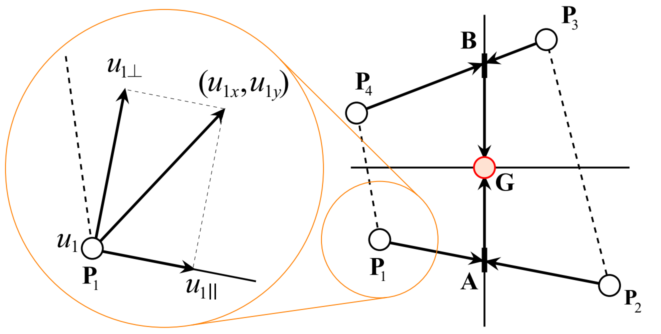


Fig. 3. Calculation of the grid velocity from the particle velocities

B. Particle-to-Grid Velocity Conversion

Suppose that each particle has moved according to the velocity $\mathbf{u}^P$, carrying the spatial derivatives of the velocity as well as the velocity itself. We must now develop a procedure to transfer the consequences of the particle advections to the grid. The particle-to-grid velocity conversion is more complicated than the grid-to-particle conversion.

Referring to Figure 3, let $\mathbf{G}$ be a grid point. From each quadrant of $\mathbf{G}$ we select the nearest particle, namely, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, and $\mathbf{P}_4$. Let $\mathbf{u}_i^P = (u_i^P, v_i^P)$ be the velocity of $\mathbf{P}_i$, and let (u_{ix}^P, u_{iy}^P) and (v_{ix}^P, v_{iy}^P) be the derivatives of the x- and y-components of $\mathbf{u}_i^P$, respectively. We must find formulae for the velocity $\mathbf{u}^G = (u^G, v^G)$ at $\mathbf{G}$ and its spatial derivatives. Because the four particles are not rectangularly located, we cannot use conventional grid-based CIP interpolation.

Our particle-to-grid velocity conversion is composed of the following steps. For simplicity, we only show the calculation of the x-components.

(1) We coordinate-transform the derivative (u_{1x}^P, u_{1y}^P) of u_1^P into $(u_{1\|}, u_{1\perp})$, so that $u_{1\|}$ represents the component along the direction $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$ and $u_{1\perp}$ represents the component perpendicular to $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$. Similarly, we obtain the coordinate-transformed derivative $(u_{2\|}, u_{2\perp})$ of u_2^P.
(2) On the results obtained in Step (1), we perform one-dimensional monotonic CIP interpolation along the direction $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$ in order to calculate u_A and its $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$-directional derivative $(u_{A\|}, u_{A\perp})$ at the location $\mathbf{A}$. $u_{A\perp}$ is just linearly interpolated because it is the

component perpendicular to $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$.[3]

(3) We obtain (u_{Ax}, u_{Ay}) from $(u_{A\parallel}, u_{A\perp})$ by applying the inverse of the above coordinate-transformation.

(4) Similarly, we perform Steps (1)~(3) for the particles $\mathbf{P}_3$ and $\mathbf{P}_4$ to calculate u_B and (u_{Bx}, u_{By}) of $\mathbf{B}$.

(5) On the results obtained in Steps (3) and (4), we perform y-directional monotonic CIP interpolation, which gives the velocity and derivative at $\mathbf{G}$.

This monotonic interpolation method can be straightforwardly extended to the 3D case. We note that the present work does not employ general radial basis interpolation schemes because (1) they do not exploit the derivative information, and (2) modifying the schemes so that they reflect the derivatives while they maintaining the monotonicity turns out to be difficult.

Preserving monotonicity is a necessity condition for stable simulation. As described in Sections IV and V, particle-to-grid conversion, grid-to-particle conversion, and octree-based CIP, all these components are monotonic. Therefore, the derivative particle model is stable. We did not run into any instability in performing the experiments shown in Section VI.

C. The Level Set Conversion

The level set conversion should be performed differently from the velocity conversion; level set values represent the minimum distances to the surface, so the conversion should not be based on interpolation. Here we use a modified version of a widely used level set conversion procedure [1], [39]; the new conversion turns out to produce more accurate results.

In the original particle level set method, a spherical implicit function $\phi(\mathbf{x}) = s_p(|\phi_p| - |\mathbf{x} - \mathbf{x}_p|)$ is used to calculate the level set value at grid point $\mathbf{x}$, where ϕ_p is the level set of the particle, and $s_p = +1\ (-1)$ for the positive (negative) particles. In the present work, utilizing the derivative information stored in the derivative particle, we calculate the level set and its derivative at the grid point with

$$\phi(\mathbf{x}) = \phi_p + \frac{\nabla\phi_p}{|\nabla\phi_p|} \cdot (\mathbf{x} - \mathbf{x}_p) \qquad (8)$$

[3] [38] states that such kind of linear interpolation does not critically disgrade the overall accuracy of the algorithm.

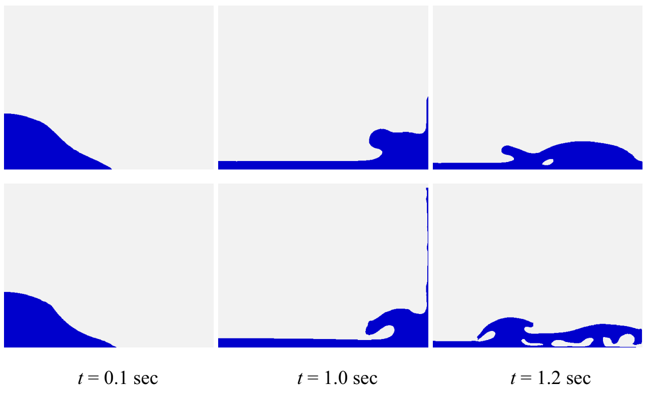


Fig. 4. Snapshots taken from 2D breaking-dam simulation: the upper and the lower sequences show the results produced from the traditional linear model with the particle level set method and the derivative particle model, respectively.

$$\nabla\phi(\mathbf{x}) = \nabla\phi_p \qquad (9)$$

where $\nabla\phi_p$ is the gradient carried by the derivative particle. Because Equation (8) is based on the gradient-directional distance rather than the Euclidean distance, it produces more accurate results than the spherical implicit function. Except for the above modifications, the conversion procedure is identical to the one presented in Enright et al. [39].

VI. Experimental Results

The technique presented in this paper was implemented on a Power Mac with Dual G5 2.5 GHz processors and 5.5 GB of memory. We used the program to simulate several experimental situations that produce high-Reynolds-number fluid behaviors in the real world. In the experiments, we used the following constants: $g = -9.8$ m/sec^2, $\rho_{water} = 1000$ kg/m^3, $\mu_{water} = 1.137 \times 10^{-3}$ kg/ms, $\rho_{air} = 1.226$ kg/m^3, and $\mu_{air} = 1.78 \times 10^{-5}$ kg/ms, where g is gravity. In all experiments, we restricted the simulation time step under the CFL number 3. Extraction of water surface was done using the marching cube algorithm, and rendering was done by mental ray®.

a) Breaking-dam: In order to compare the numerical damping between the derivative particle model and the linear model with the particle level set method, we performed 2D breaking-dam test with 128^2 effective resolution, in which 0.2×0.4m water column was released in the gravity field. Figure 4 shows the results. We can see that derivative particles produce less diffusive result: breaking of the wave is sharper and vorticities are well conserved.

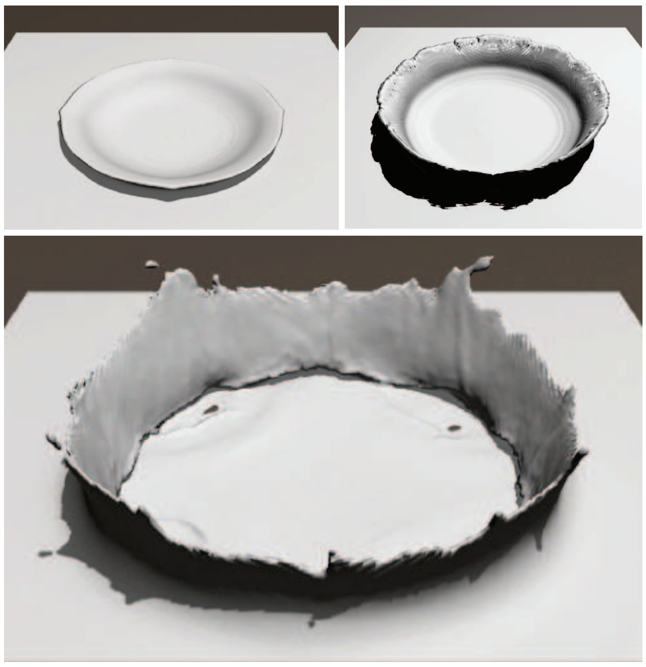


Fig. 5. Snapshots taken from the water drop simulation: the top left and top right images show the results produced from the traditional linear model with the particle level set method and the CIP method, respectively. The bottom image shows the result from the derivative particle model.

b) A Chunk of Water Dropping onto Shallow Water: Figure 5 shows a snapshot taken from the simulation in which a chunk of water is dropped onto a 0.05-m deep water reservoir. We used the no-slip boundary condition for the bottom surface of the reservoir, which caused the water to move fast at the top but slow at the bottom, resulting in a crown-like splash. This experiment was performed with an effective grid resolution of 256^3. The same scene was also simulated with the traditional linear model that uses the particle level set method and with the grid-based CIP method. The comparison demonstrates that the proposed technique produces a more realistic result. The derivative particle model took 60 sec per time-step in average, the linear model took 30 sec, and the CIP method took 48 sec. all including the file output time.

c) Impact of a Solid Ball: In this experiment, a solid ball with radius 0.15 m makes an impact into water, with the velocity $(5.0, -3.0, 0.0)$ m/sec. The impact generates complex structures shown in Figure 1. This experiment manifests that the proposed technique can produce detailed fluid movements and surface features that occur in violent solid-water-air interactions. The effective resolution of this experiment was $192 \times 128 \times 128$.

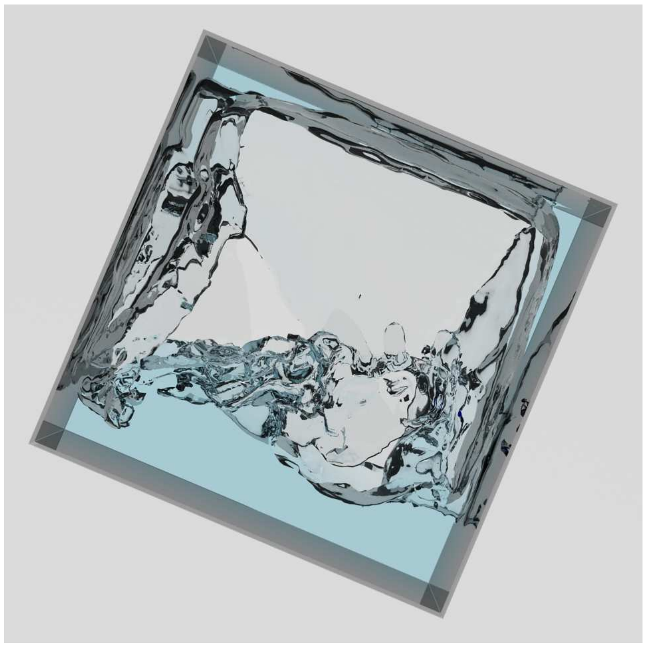


Fig. 6. A snapshot taken from the simulation of a rotating water tank.

d) Rotating Water Tank: Figure 6 shows a $0.9 \times 0.9 \times 0.3$ m rotating box with half-full of water. In the experiments, the centrifugal force creates the effect of pushing the water to the side of the tank. This simulation was performed with an effective grid resolution of $96 \times 96 \times 32$. This experiment demonstrates that the derivative particle model can simulate complex movements of fluids even in a coarse-resolution grid.

A. Velocity-Dissipation Suppression Test

To test the performance of the derivative particle model in regard to velocity-dissipation suppression, we performed the experiment described below. For the comparison, we also performed the same experiment with the FLIP method. We set up a 2D $1m \times 1m$ square box with 64^2 uniform grids. We initially set the grid velocities to the incompressible velocity field defined by the stream function [40]

$$\psi = \frac{1}{\pi} sin^2(\pi x) sin^2(\pi y). \tag{10}$$

The velocities of the particles were obtained with the method described in Section V-A. After advecting the particles and converting particle velocities to grid velocities, we projected the velocities in order to maintain incompressibility. The above procedure

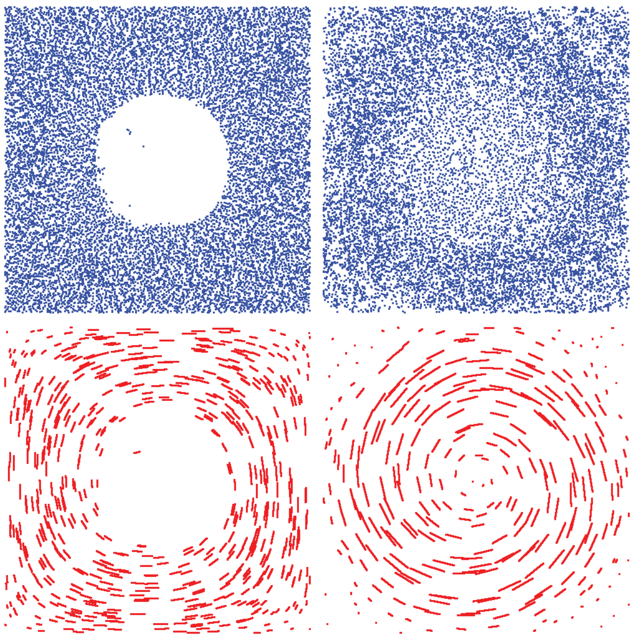


Fig. 7. 2D vorticity preservation test at $t = 5$ sec. The left and the right columns are the results from the FLIP method and the derivative particle model, respectively. The top images show the particles' position, and the bottom images show the particles' velocity. The length of the red lines are proportional to the magnitude of the particles' velocities.

was repeated for every time step. The same number of particles were used for both FLIP and derivative particle experiments. We didn't perform any particle reseeding step. Fig 7 compares the results; the images in the left column are from FLIP and the images in the right column are from derivative particles model. FLIP generated a hole in which no particle exists at $t = 0.9$ sec, and the hole expanded afterwards. In the bottom-row images of Fig 7, we can clearly see that the velocity damping of the derivative particles model is smaller than that of the FLIP. We used Zhu and Bridson's code[4] for the FLIP simulation.

B. *Volume Conservation Test*

To see the mass conservation performance of the proposed derivative particle level set conversion(Equation (8) and (9)), we performed 3D incompressible spiral field test [39]. A sphere of radius 0.15 is set initially at $(0.35, 0.35, 0.35)$. The velocity field is adopted from [39]. The effective resolution is 256^3 grid cells. For the comparison, the same test was done also with the particle level set method. Figure 8 shows the results. When the sphere was maximally stretched, the particle level set method has lost 4.454% of the initial volume while the derivative particle level set conversion has lost 3.151% of the initial volume. After one complete revolution, the particle level set method has lost 7.460%, while the proposed method has lost 3.447%.

[4]http://www.cs.ubc.ca/˜rbridson/

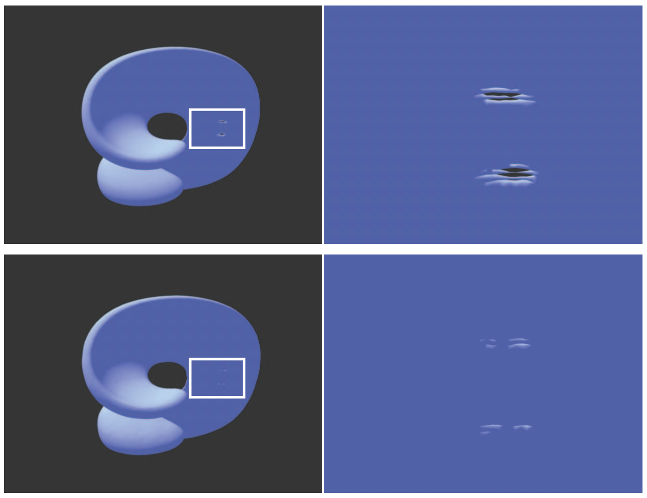


Fig. 8. Snap shots from 3D incompressible spiraling field test at $0.52T$. The top left and bottom left images show the results from the particle level set method and the proposed derivative particle level set conversion, respectively. The images in the right column are magnifications of a rectangular part in those experiments.

VII. Conclusion

In this paper, we have introduced a new concept called the *derivative particle*, and proposed a fluid simulation technique that increases the accuracy of the advection part. The non-advection part of the simulator is implemented using the Eulerian scheme, whereas the advection part is implemented using the Lagrangian scheme. A nontrivial problem in developing simulators that combine schemes in this way is how to convert physical quantities at the switching of the advection and non-advection parts. We successfully overcame this problem by developing conversion procedures that effectively suppress unnecessary numerical dissipation. In the fluid regions where the particle advection does not need to be performed, the advection is simulated using an octree-based CIP solver, which we developed in this work by combining CIP interpolation with the octree data structure. This approach also served to reduce numerical dissipation. The results of several experiments demonstrated that the proposed technique significantly reduces velocity dissipation, leading to realistic reproduction of dynamic fluids.

The present work would not have been possible without the pioneering work of previous researchers. The idea of entirely using particles to solve the advection part came from the PIC [37] and FLIP [31] methods, but for the particle-to-grid and grid-to-particle velocity conversion we used derivative-based cubic interpolation instead of the linear interpolation. In addition, the notion of letting the particles carry the level set values came from the particle level set method [1]. This model was extended in the present work so that particles also carry the velocity information. Moreover, utilizing the derivative information came from the CIP method [36]. Here we extended the application of this method to particles, which entailed the development of a particle-to-grid velocity conversion procedure that contained an extension of the CIP method itself: the CIP interpolation of the particles in non-rectangular configurations.

In the experiments, the ability of reproducing the *details* was emphasized. But the proposed technique can also apply to large scale fluid movements to produce accurate results. The technique involves increased amount of memory for storing the derivative information. However, we demonstrated that a 256^3 grid can be simulated on a contemporary PC, which is already a viable resolution for portraying interesting fluid scenes.

ACKNOWLEDGMENT

This work was supported in part by Ministry of Science and Technology under National Research Laboratory (NRL) grant M10600000232-06J0000-23210, Ministry of Information and Communication, the Brain Korea 21 Project, the faculty research fund of Sejong University in 2006, and Automation and System Research Institute at Seoul National University.

REFERENCES

[1] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 736–744, 2002.

[2] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 457–462, 2004.

[3] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graphical models and image processing: GMIP*, vol. 58, no. 5, pp. 471–483, 1996.

[4] ——, "Controlling fluid animation," in *Computer Graphics International 97*, 1997, pp. 178–188.

[5] J. Stam, "Stable fluids," *Computer Graphics (Proc. ACM SIGGRAPH '99)*, vol. 33, no. Annual Conference Series, pp. 121–128, 1999.

[6] A. Staniforth and J. Côtè, "Semi-lagrangian integration scheme for atmospheric model - a review," *Mon. Weather Rev.*, vol. 119, no. 12, pp. 2206–2223, 1991.

[7] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large scale phenomena," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 703–707, 2003.

[8] B. E. Feldman, J. F. O'Brien, and O. Arikan, "Animating suspended particle explosions," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 708–715, 2003.

[9] A. Treuille, A. McNamara, Z. Popović, and J. Stam, "Keyframe control of smoke simulations," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 716–723, 2003.

[10] A. McNamara, A. Treuille, Z. Popović, and J. Stam, "Fluid control using the adjoint method," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 449–456, 2004.

[11] B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 904–909, 2005.

[12] B. E. Feldman, J. F. O'Brien, B. M. Klingner, and T. G. Goktekin, "Fluids in deforming meshes," in *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2005*, 2005, pp. 255–259.

[13] N. Foster and R. Fedkiw, "Practical animation of liquids," *Computer Graphics (Proc. ACM SIGGRAPH 2001)*, vol. 35, pp. 23–30, 2001.

[14] D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-lagrangian particle level set method," *Computers and Structures*, vol. 83, pp. 479–490, 2005.

[15] M. Carlson, R. J. Mucha, and G. Turk, "Rigid fluid: Animating the interplay between rigid bodies and fluid," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 377–384, 2004.

[16] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw, "Coupling water and smoke to thin deformable and rigid shells," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 973–981, 2005.

[17] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien, "A method for animating viscoelastic fluids," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 463–468, 2004.

[18] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 965–972, 2005.

[19] J.-M. Hong and C.-H. Kim, "Discontinuous fluids," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 915–920, 2005.

[20] H. Wang, P. J. Mucha, and G. Turk, "Water drops on surfaces," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 921–929, 2005.

[21] J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," *Computer Graphics (Proc. ACM SIGGRAPH '95)*, vol. 29, no. Annual Conference Series, pp. 129–136, 1995.

[22] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003*, 2003, pp. 154–159.

[23] M. Müller, B. Solenthaler, R. Keiser, and M. Gross, "Particle-based fluid-fluid interaction," in *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2005*, 2005, pp. 237–244.

[24] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker, "Particle-based simulation of fluids," in *Eurographics 2003 proceedings*. Blackwell Publishers, 2003, pp. 401–410.

[25] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of

smoke," *Computer Graphics (Proc. ACM SIGGRAPH 2001)*, vol. 35, pp. 15–22, 2001.

[26] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 910–914, 2005.

[27] S. I. Park and M. J. Kim, "Vortex fluid for gaseous phenomena," in *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2005*, 2005, pp. 261–270.

[28] B. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Flowfixer: Using bfecc for fluid simulation," in *Eurographics Workshop on Natural Phenomena 2005*. Blackwell Publishers, 2005.

[29] T. F. Dupont and Y. Liu, "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function," *J. Comp. Phys.*, vol. 190, pp. 311–324, 2003.

[30] O.-Y. Song, H. Shin, and H.-S. Ko, "Stable but non-dissipative water," *ACM Transactions on Graphics*, vol. 24, no. 1, pp. 81–97, 2005.

[31] J. U. Brackbill and H. M. Ruppel, "Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions," *J. Comp. Phys.*, vol. 65, pp. 314–343, 1986.

[32] X.-D. Liu, R. Fedkiw, and M. Kang, "A boundary condition capturing method for poisson's equation on irregular domains," *J. Comp. Phys.*, vol. 160, pp. 151–178, 2000.

[33] M. Kang, R. Fedkiw, and X.-D. Liu, "A boundary condition capturing method for multiphase incompressible flow," *J. Sci. Comput.*, vol. 15, pp. 323–360, 2000.

[34] T. Yabe and T. Aoki, "A universal solver for hyperbolic equations by cubic-polynomial interpolation i. one-dimensional solver," *Comp. Phys. Comm.*, vol. 66, pp. 219–232, 1991.

[35] ——, "A universal solver for hyperbolic equations by cubic-polynomial interpolation ii. two- and three dimensional solvers," *Comp. Phys. Comm.*, vol. 66, pp. 233–242, 1991.

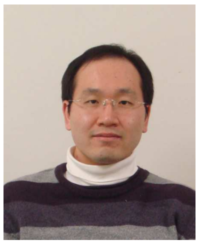
[36] T. Yabe, F. Xiao, and T. Utsumi, "The constrained interpolation profile method for multiphase analysis," *J. Comp. Phys.*, vol. 169, pp. 556–593, 2001.

[37] F. H. Harlow, "The particle-in-cell method for numerical solution of problems in fluid dynamics," in *Experimental arithmetic, high-speed computations and mathematics*, 1963.

[38] T. Yabe, H. Mizoe, K. Takizawa, H. Moriki, H.-N. Im, and Y. Ogata, "Higher-order schemes with cip method and adaptive soroban grid towards mesh-free scheme," *J. Comp. Phys.*, vol. 194, pp. 57–77, 2004.

[39] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A hybrid particle level set method for improved interface capturing," *J. Comp. Phys.*, vol. 183, pp. 83–116, 2002.

[40] J. Bell, P. Colella, and H. Glaz, "A second-order projection method for the incompressible navier-stokes equations," *J. Comp. Phys.*, vol. 85, pp. 257–283, 1989.



Oh-young Song received the BS, MS, and PhD degrees in the School of Electrical Engineering and Computer Science from Seoul National University (SNU), Korea, in 1998, 2000, and 2004, respectively. From 2004-2006, he was a postdoctoral fellow in the School of Electrical Engineering and Computer Science at SNU. Since 2006, he has been an assistant professor in the Department of Digital Contents at Sejong University, Korea. His primary research interests are physics-based animation and character animation.

Doyub Kim received the BS degree in the School of Electrical Engineering and Computer Science from Seoul National University (SNU), Korea, in 2005. He is currently working toward the PhD degree. His research interests include fluid simulation, cloth animation, and rendering.

Hyeong-Seok Ko received the BS and MS degrees in computer science from SNU, Korea, in 1985 and 1987, respectively, and PhD degree in computer science from University of Pennsylvania in 1994. He is a professor in the School of Electrical Engineering and Computer Science at Seoul National University (SNU), Korea. He has been working at SNU since 1994. His recent research interest includes cloth simulation, fluid simulation, facial animation, hair modeling, flexible body animation, and motion retargeting.