Computers & Graphics 36 (2012) 1025–1035

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

VEA 2012

Real-time ink simulation using a grid-particle method

Shibiao Xu, Xing Mei, Weiming Dong, Zhiyi Zhang, Xiaopeng Zhang*

National Laboratory of Pattern Recognition, NLPR-LIAMA, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 28 March 2012
Received in revised form
15 August 2012
Accepted 15 August 2012
Available online 25 September 2012

Keywords:
Ink diffusion
Particle-guided
Coupling
Drag force
Motion blur
Multiple ink
Animation

ABSTRACT

This paper presents an effective method to simulate the ink diffusion process in real time that yields realistic visual effects. Our algorithm updates the dynamic ink volume using a hybrid grid-particle method: the fluid velocity field is calculated with a low-resolution grid structure, whereas the highly detailed ink effects are controlled and visualized with the particles. To facilitate user interaction and extend this method, we propose a particle-guided method that allows artists to design the overall states using the coarse-resolution particles and to preview the motion quickly. To treat coupling with solids and other fluids, we update the grid-particle representation with no-penetration boundary conditions and implicit interaction conditions. To treat moving "ink-emitting" objects, we introduce an extra drag-force model to enhance the particle motion effects; this force might not be physically accurate, but it proves effective for producing animations. We also propose an improved ink rendering method that uses particle sprites and motion blurring techniques. The simulation and the rendering processes are efficiently implemented on graphics hardware at interactive frame rates. Compared to traditional fluid simulation methods that treat water and ink as two mixable fluids, our method is simple but effective: it captures various ink effects, such as pinned boundaries (Nelson, 2005 [1]) and filament patterns (Shiny et al., 2010 [2]), while still running in real time, it allows easy control of the animation, it includes basic solid–fluid interactions, and it can address multiple ink sources without complex interface tracking. Our method is attractive for animation production and art design.

1. Introduction

Ink is unique among all types of pigments: different concentrations of ink exhibit different intensity levels, and for all concentrations, intensity levels depend on contact time [3]. Solid ink is a mixture of soot and glue, and it is ground together with water [4]. The most attractive visual characteristic of liquid ink is its dynamic diffusion process when it is dropped into water: it spreads out in the water, and the mixture of the ink and the water molecules looks delicate and beautiful. This phenomenon is of great interest for art production. In this paper, we intend to simulate realistic visual effects of dynamic ink diffusion with high performance and apply these special effects in animation production.

When liquid ink drops spread in water, dynamic effects occur not only around the fluid interface but also inside the fluid volume. It is costly or sometimes impossible to capture these real-world special effects in some conditions. Therefore, synthetic methods have been extensively used in computer animation industry, and most of these methods calculate the ink diffusion process using fluid simulation techniques.

Current fluid simulations can be classified into two primary categories: Eulerian methods and Lagrangian methods. Eulerian methods [5–7] utilize a grid at fixed points in space and calculate fluid quantities in the grid. Instead of using fixed points, Lagrangian methods [8–10] treat the fluid as a particle system. Both methods have limitations. Simulations that use a three-dimensional computational grid have high computational costs, so it is difficult to simulate fine details and large-scale phenomena. Particle methods such as smoothed particle hydrodynamics (SPH) [11,12] are memory efficient, but they face other difficulties such as finding neighbors and tracing the interface between different fluids. Recently, many hybrid methods [13,14] have been proposed to accurately simulate fluids and resolve fine details. Our method is a novel hybrid approach to simulating dynamic ink diffusion effects.

Due to the significant advancement of fluid simulation, mixable fluid phenomena have become a popular topic of research, and multiple fluid simulations often model ink diffusion in water, for example. However, these methods primarily focus on the interfaces of different fluids and simulate the multiple fluids in a general manner, and they do not consider the attractive effects observed when ink diffuses in water, so they cannot simulate small internal movements and small-scale features, such as

* Corresponding author. Tel.: +86 62650316; fax: +86 10 62647458.
E-mail address: xpzhang@nlpr.ia.ac.cn (X. Zhang).

0097-8493/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2012.08.003

filament patterns [2]. Furthermore, the methods are computationally expensive. Therefore, the methods cannot be utilized directly for animation production, which requires results with a realistic visual appearance, fast feedback and low computational expense.

When simulating the small-scale features of ink, even tiny movements will have significant effects on the results, so it is beneficial if the animation can be efficiently simulated, edited and even stored by artist interactively. Currently, there are several three-dimensional computer graphics software packages, such as SideFX Houdini and Autodesk Maya, that are used to create computer animations; however, most of the programs are limited, especially for simulating fluid phenomena. There are some limitations that should be improved to better simulate fluids. First, because the packages are general three-dimensional graphics software, they simulate fluids in common without considering any special effects. Second, the time step cannot be too large; otherwise the simulation will be unstable, so the simulations cannot be run in real time. Therefore, the simulations require significant time, and it is difficult to freely simulate the special effects of ink diffusion. Fig. 1 shows an example of ink diffusion in water where we have used Maya fluids for the simulation. Artists must adjust the opacity property of the fluids repeatedly and increase the key frames in Maya, so it is impossible to obtain more internal details of the ink flow. Furthermore, in this traditional method, the emitters must be static and the ink flow cannot interact with other objects; otherwise the results will be artificial, so we cannot apply this method in computer animation.

In this paper, we propose simulating dynamic ink diffusion using an improved hybrid grid-particle method [15]. Fig. 2 shows an example in which ink particles are driven by water. In Fig. 2(a), there are some words written with ink particles. In Fig. 2(b), ink particles are advected by the water. Based on the components of real liquid ink, we use particles to represent the ink particles and use a grid to represent water dissolving the ink particles. In fact, the water-containing grid cells and ink particles are emitted from the ink sources simultaneously. Our simulation visualizes the highly detailed ink effects with particles only by assuming the water grid cells are completely transparent, but the region of

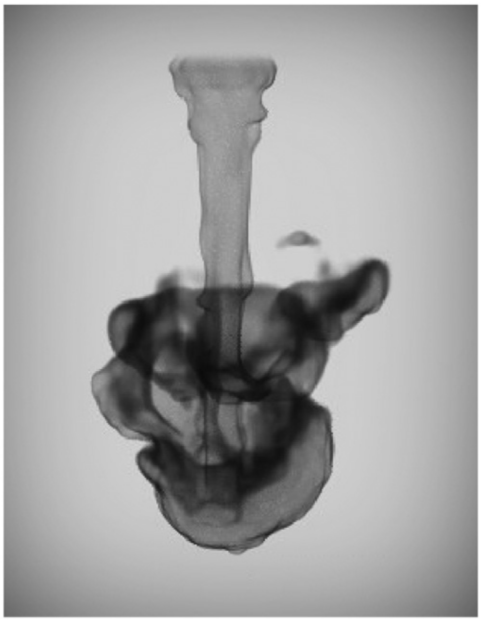

Fig. 1. An example of ink diffusion in water using Maya fluids. The internal opacity of ink flow is lower than the external value.

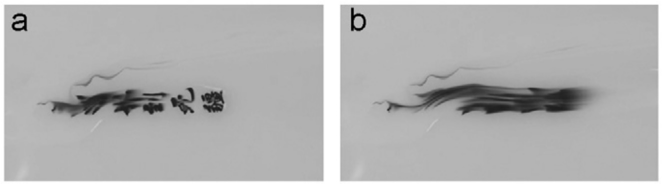

Fig. 2. An example that ink particles are driven by the water. (a) Words written with ink. (b) Water advect ink particles.

water grid cells controls the advection of the particles and sets the boundary conditions. Thus, all the spatial interactions, including coupling, incompressibility and pressure, are calculated using a low-resolution grid structure, and the ink particles are only used to treat advection, which depends on the water's velocity field.

Controlling the motion and appearance of fluid is necessary for artistic effects that do not exist in the real world; however, the grid-based method does not permit user interaction during the simulation. Therefore, we propose a particle-based method to extend the hybrid grid-particle representation above: we can use small number of large particles to specify the initial states and preview the motion quickly. The particle-based method is very effective and easy to implement. It enables user interaction during the simulation and allows the user to easily control the fluid motion as desired. In ink animation, it is necessary to couple solids and fluids, so we update the grid-particle method with no-penetration boundary conditions and implicit interaction conditions to address solid–fluid interactions and multiple ink sources; our method can naturally preserve the complex geometry nature of the interface of different inks without complex interface tracking. For objects moving at high speeds, we introduce an extra drag force to enhance the effects of particle motion, and it is useful to create realistic ink effects when producing animations. Drag is a force that opposes the motion of moving objects. Here, it is used to apply a force to ink particles in the direction opposite to that of the particles' movement; otherwise, the particle motion effects will be artificial. The rendering of ink particles is crucial because every point is rendered, and the result will likely appear grainy if an insufficient number of particles are used. Solving this problem requires a large number of particles. However, using more ink particles results in increased computation time. To keep the details realistic and smoothly varying in real time, we propose a novel ink rendering method that uses the sprite rendering and motion blur techniques.

The simulation is implemented on graphics hardware in real time. Compared to traditional simulation methods that incur greater computational costs, our method is simple and effective. In our simulation, the internal details of liquid ink are equally as important as the interface; therefore, our method can capture various ink effects, such as pinned boundaries and filament patterns. Our method allows the artist to easily interact with the simulation; thus, the ink animation can be efficiently simulated, edited and even stored by artist in real time. Because our method satisfies the no-penetration boundary conditions, which allow the fluid to tangentially slip past but not flow through the solid, it can treat the coupling between the ink and a dynamic solid moving at high speed, and the velocity of ink particles will change over time because of interactions with the dynamic solid. Our method can solve for the interactions among multiple ink sources without using complex interface tracking. Regarding the simulation of ink–ink coupling, different ink particles are controlled by the same water grid cells and there are implicit interaction conditions between different ink particles, so our method can naturally identify the interface of different fluids without performing any additional calculations. Our method can be used for practical applications in the animation industry.

Our main contributions can be summarized as follows:

- A particle-guided user interaction method to specify the initial states easily and quickly preview the large-scale motion of ink flows.
- An improved hybrid grid-particle method with implicit interaction conditions to simulate multiple fluids and generate the complex interfaces between different inks flow naturally.
- A drag force model for "ink-emitting" objects moving at high speeds in practical animations.

2. Related work

We briefly review several papers relevant to our work.

General fluid simulations: Physically based fluid simulations are often performed using the incompressible Navier–Stokes equations. Foster and Metaxas [16] first introduced a fully three-dimensional Navier–Stokes solver for computer graphics fluid simulations. Their method works well, but the explicit time-stepping scheme can result in instabilities. Stam [17] improved this approach tremendously by presenting a semi-Lagrangian method for solving the Navier–Stokes equations that is unconditionally stable for any time step size. Because the method is stable even when large time steps are used and the linear interpolation is first-order, the semi-Lagrangian velocity advection intrinsically includes dissipation, which may remove small-scale and even large-scale motions. In 2001, vorticity confinement [18] was added to generate small-scale fluid rolling motion. Vortex filaments [19,20] provide a technique to capture fine-scale vortices, and they define the fluid velocity using a series of filaments of vorticity. Their method differs from traditional momentum-based particle and grid simulations because each vortex filament contributes velocity to the entire simulation space rather than just a localized region and each vortex filament is advected using the sum of the velocity contributions from all the other filaments. Therefore, the method increases the computational complexity of the simulation. Back and forth error compensation and correction (BFECC) [21] can also be applied to reduce the dissipation and diffusion that occurs at multiple steps of the advection calculation, such as when the velocity and density are calculated. Furthermore, there are many higher-order methods that can be used to improve the accuracy of the advection calculation; however, the BFECC method can be implemented more easily and exhibits second-order accuracy in both space and time. Recently, a wide variety of methods, including Eulerian methods [5–7] and Lagrangian methods [8–10] have been developed. Compared to Lagrangian methods, Eulerian methods are easier to analytically apply to spatial derivatives, such as the pressure gradient and the viscosity term, but they have difficulties in the advection term with large time steps. Therefore, the particle-in-cell (PIC) method [22] and the fluid-implicit-particle (FLIP) method [23] are proposed to treat advection with particles instead of a grid. The most significant problem with the PIC and FLIP methods is the excessive numerical diffusion caused by averaging the fluid variables repeatedly. Zhu and Bridson [14] developed a fluid solver that combines the PIC method and the FLIP method to generate sand animation efficiently; in their method, only particles are used to represent the fluid, and the grid is for auxiliary calculations. We also combine the approaches of grids and particles to perform realistic ink simulations.

Multiple fluid simulations: Because of significant advances in fluid simulation technology, simulating the interactions of multiple fluids has been a popular topic of research in the graphics field. Zhu [24,25] simulated a pair of miscible fluids using a lattice Boltzmann method (LBM); however, when the Reynolds number is too high, LBM is unstable and cannot satisfy the divergence-free condition of fluids. In addition, LBM-based approaches cannot be easily integrated into conventional fluid simulation pipelines. Bao [26] proposed a novel volume fraction based approach that can treat both immiscible and miscible interactions between fluids well and yield much finer details of fluid mixing. However, to handle interactions involving multiple fluids with different properties, they use special methods that are too difficult and complex to be applied in practical animation applications in real time. In our simulation, we use the semi-Lagrangian method of Stam [17], which is the standard operator-splitting method for solving the Navier–Stokes equations to obtain a divergence-free fluid, vorticity confinement [18] and the BFECC method [21] instead of traditional first-order advection to mitigate the numerical dissipation of the semi-Lagrangian method.

Nahyup and Jinho [27] presented a hybrid approach to simulating multiple fluids by combining distance functions (level set functions) and volume fractions. They also adopt the standard operator-splitting method [17] to solve the Navier–Stokes equations on a grid. To resolve the interfaces between multiple fluids, they must modify the volume fractions of cells near the interfaces at each time step. Furthermore, the material diffusion equation is required for modeling diffusion phenomena using volume fractions. Seung-Ho and Hyeong [2] used a hybrid method that combines a grid-based method and smoothed particle hydrodynamics to simulate molecular diffusion and attraction. They primarily focused on the small-scale phenomena that occur when a fluid spreads out at the interface of another fluid, which cannot describe the filaments inside the fluid. In their simulations, they make two assumptions: one is that the mixing process is equivalent to a fluid passing through a porous medium according to Darcy's Law, and the other is that the mixing surface has a fractal-like shape. In summary, both the approaches above cannot simulate small-scale features such as the filaments of ink diffusion. They also must treat the complex interfaces between every pair of fluids, and their methods are computational expensive because they must consider many additional physical equations and assumptions; thus, the methods cannot be used effectively for computer animations.

Ink simulations: Several authors have researched simulating ink effects, but most focused on the dispersion of ink in the paper. Guo and Kunii [28] proposed the first model designed specifically to simulate ink dispersion in the paper by using particles of different sizes. Many other methods to simulate the dispersion phenomenon, including the partial differential equations (PDE) method [3] and cellular automation method [29], have been proposed. However, these methods are only suitable for situations where ink is mixed to form blurry images without the interesting flow patterns observed in reality. Nelson [1] presented a physically based method for simulating ink dispersion in absorbent paper that can simulate complex and realistic ink effects. They implemented a digital paint system that can simulate various effects of ink dispersion on the paper in real time.

Xu [15] proposed a hybrid grid-particle method based on the components of liquid ink that can efficiently generate the special effects of ink diffusion in real time; however, there are also several problems with attempting to use this hybrid approach in industrial animation applications where artists must be able to interact with the simulation. Furthermore, the method was designed primarily for single-ink flows, so it cannot simulate multiple-ink flows directly. Because water grid cells and ink particles can be emitted from various types of sources, the possibility that the sources are dynamic cannot be ignored. Thus, ink particles should face resistance from the external fluid field; otherwise, the ink flow will be artificial.

3. Ink diffusion simulations

Diffusion exists in all states of matter, solid, liquid, and gas, and refers to the movement of molecules from a region of high concentration to a region of lower concentration. However, diffusion only occurs rapidly enough to be observable in liquids and gases. We describe the special effects of real ink diffusion in water and then employ a novel hybrid grid-particle representation [15] based on the visual characteristics of ink diffusion to simulate this phenomenon.

3.1. Visual characteristics of ink diffusion

Solid ink is a mixture of soot and animal glue [4]. The solid is ground together with water to yield black liquid ink [1]. The soot is composed of carbon particles and dissolves easily in water, and the animal glue is used to control the diffusion of the ink flow. The process of diffusion is delicate and beautiful; to better appreciate this type of diffusion, we provide a video recording of the diffusion process, and some frames are shown in Fig. 4.

To perform a realistic simulation, we must include the essential effects of real diffusion, which include the following:

Pinned boundaries [1] (Fig. 4(c)): As liquid ink is dropped into water, the high-density ink particles are quickly diluted and dissolved by the water. Because of surface tension and the water solubility of ink, the liquid ink's boundary is pinned by the water that dissolves the ink particles and rapidly becomes ring-like, which means the ink particles cannot flow out the water region. Furthermore, if there is another type of ink particle in another region, those particles will also be pinned outside the water region until they are dissolved gradually by the water. This effect is more prominent for concentrated ink, which displays a darkening of the boundary.

Filament patterns [2] (Fig. 4(d)): When ink particles diffuse in water, they spread outward in every direction to regions of low concentration. The transparency reduces over time, and the texture becomes as homogeneous as a piece of silk. Delicate ink streaks that follow the direction of the water flow will appear gradually. The streaks appear because some of the ink dissolves in water and the water molecules control the advection of the ink molecules.

3.2. Overview of the method

The construction of a liquid ink model includes specifying the components of the ink and the construction of the flow mechanism in two parts. To accurately model the visual characters of ink diffusion in the simulation, the computing method must resolve fine details without causing excessive numerical dissipation. Another requirement is that the complex geometry characters of the interface between different liquid inks must be preserved naturally. Furthermore, the method should facilitate user interaction.

We combine Eulerian and Lagrangian methods to define our hybrid grid-particle representation as shown in Fig. 3. The representation consists of four components: a particle-guided method for user interaction; a GPU-optimized Eulerian incompressible Navier–Stokes solver with vorticity confinement and BFECC advection; a GPU-optimized Lagrangian particle simulation with a drag force; and a hardware particle sprite rendering system with motion blur. There are two types of particles in our simulation: the coarse particles that are the guiding particles and the refined particles that are used to simulate ink particles. At first, we use the coarse resolution (a small number of large particles) to simulate the overall motion and quickly preview it. Then, the entire acceleration field is applied to the grid simulation. Unlike the method of Zhu and Bridson [14], both particles and grid cells represent fluids in our simulation: the refined particles are used to represent the ink molecules, and the corresponding grid cell represents water molecules. The water grid cells and ink particles are emitted from the same virtual objects simultaneously and then mix together. Ink particles are advected along the water's velocity field such that they are maintained in the region of the water grid cells. When producing practical animations, we are only concerned with the diffusion

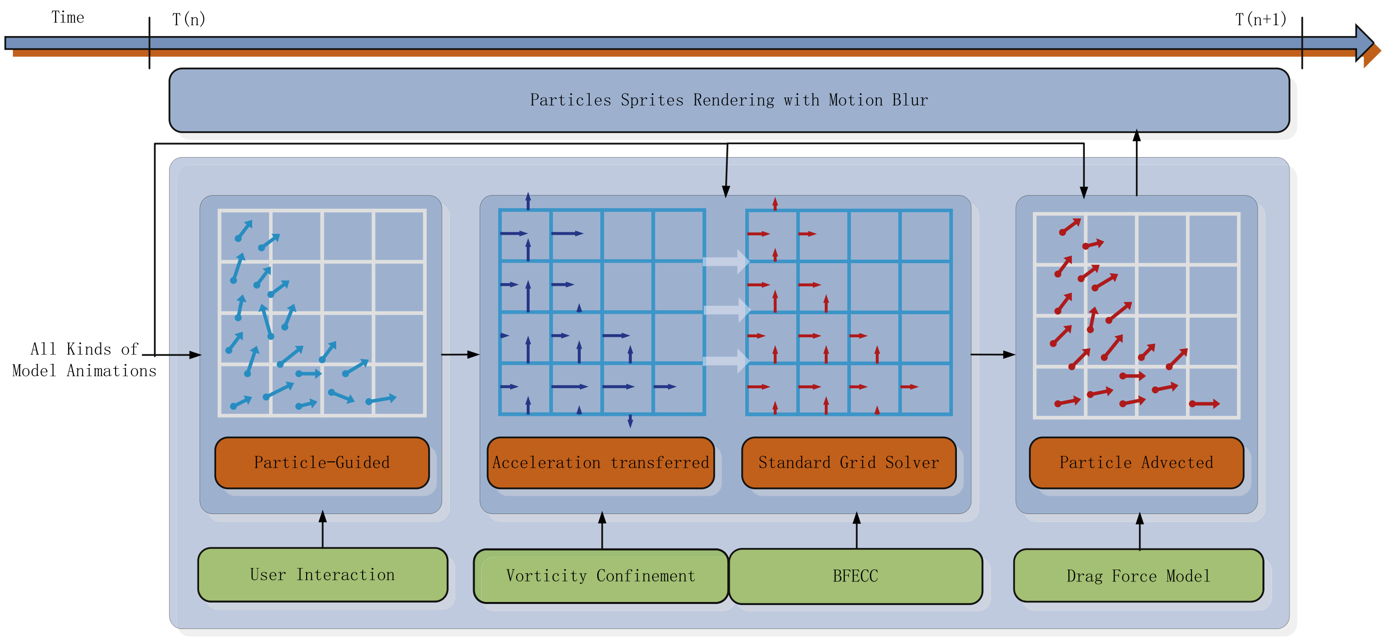

Fig. 3. One simulation iteration using our method: a particle-guided interactions for user; a GPU-optimized Eulerian incompressible Navier–Stokes solver with vorticity confinement and BFECC; a GPU-optimized Lagrangian particles simulation with drag force model; a hardware particle sprites rendering system with motion blur.

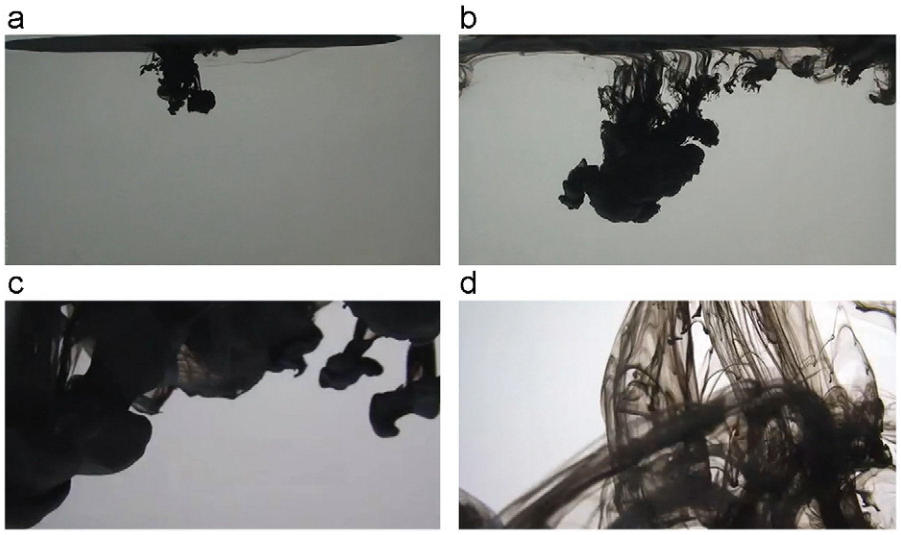

Fig. 4. The process and visual effects of real ink diffusing in water. (a) Ink drops into the water. (b) Ink spreads out in the water. (c) Pinned boundary character. (d) Filament pattern character.

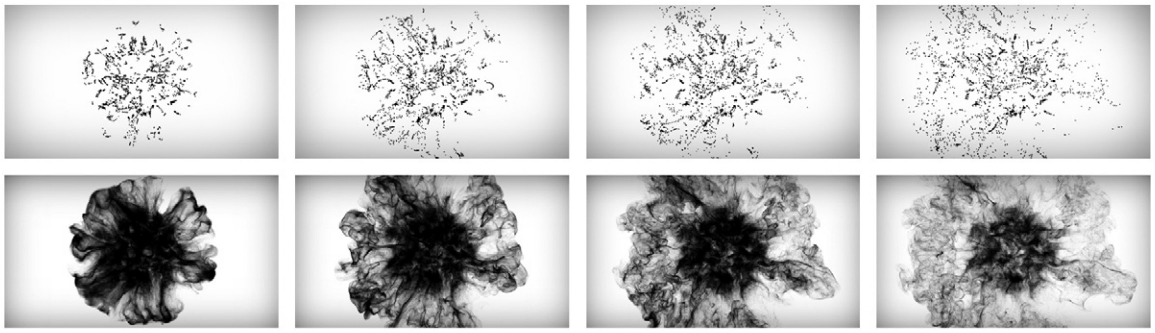

Fig. 5. An example of ink diffusing animation computed with our particle-guided method, ink flow spreads outward in every direction.

effect of ink particles, so we assume that the water is completely transparent. Thus, there is no need to render the water in our simulation, which saves hardware resources.

3.3. Particle guidance

In this paper, we define guiding as follows: an input acceleration field depending on guiding particles, $\mathbf{F}_{particle}$, is used to determine the large-scale motion of the fluid flow. The input acceleration field can be calculated using a physically based simulation or can be constructed through user interaction. Therefore, our algorithm begins with the coarse particles already assigned values for the acceleration (and any other fluid variables required). These particles are only used as necessary, and different regions of particles can be affected by different external forces during the simulation. Once the user is satisfied with the large-scale motion of the fluid flow, the acceleration field can be transferred to the grid simulation as an external force. Fig. 5 shows an example of an ink diffusion animation computed with our particle-guided method: the upper panel shows the guiding particles that are constructed through a user interaction, and the lower panel shows the ink flow spreading outward in every direction.

Ideally, the guiding should transfer the acceleration value to every fixed point on the grid. Therefore, we transfer the particle variables to a background grid by using weighted averages, as Fig. 6 shows.

3.4. Improved grid-particle method

We employ incompressible Navier–Stokes equations to simulate water on a grid with vorticity confinement as follows:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{F}_{particle} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

$$\mathbf{f}_{vort} = \varepsilon(\boldsymbol{\Psi}\times\mathbf{w})\Delta x \quad \left(\boldsymbol{\Psi} = \frac{\boldsymbol{\eta}}{|\boldsymbol{\eta}|}, \boldsymbol{\eta} = \nabla|\mathbf{w}|\right) \tag{3}$$

where $\mathbf{u}$ is the fluid velocity, ρ is the fluid density, υ is the kinematic viscosity, ε is the strength of the vorticity, $\mathbf{f}_{vort}$ is the vorticity confinement, $\mathbf{w} = \nabla\times\mathbf{u}$ is the divergence of the velocity, and $\mathbf{F}_{particle}$ represents the corresponding particle-guided forces. The numerical simulation of Eqs. (1) and (2) must be stable. Thus, we employ the standard operator-splitting method of Stam [17] to compute the velocity field on the grid.

In reality, fluid motions typically include significant rotational flows, which are termed vorticity. However, the excessive numerical dissipation in the simulation damps out these internal details of the fluids. Therefore, we apply the vorticity confinement equation (3) to restore these interesting motions. Because of the classic vector calculus identity $\nabla\cdot\nabla\times = \nabla\times\nabla\cdot$, the new velocity field with vorticity confinement is still divergence-free. Furthermore, we solve Eqs. (1) and (2) in the advection step using the second-order BFECC method instead of the semi-Lagrangian methods because the BFECC method preserves details well, is stable for any time step size, is trivially parallelizable and is efficient on GPUs.

In our simulation, we calculate velocity, pressure and other quantities at cell centers; it is easy to store the variables as textures on the GPUs. For example, the variable for velocity is denoted $\mathbf{u}_{i,j,k}$; thus, $\mathbf{u}_{i,j,k}$ is located at $(i\delta x, j\delta x, k\delta x)$, where δx is the cell spacing along x-, y-, and z-axes. This choice is only one of the methods to discretize the simulation domain; the staggered grid method also works. To enable the use of hardware that does not permit three-dimensional floating-point textures, we tile the three-dimensional texture to a two-dimensional texture. In our simulation, we apply the BFECC method in the advection step. For a given velocity field $\mathbf{u}$, it satisfies the advection equation:

$$\mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = 0 \tag{4}$$

Let L be the first-order semi-Lagrangian integration steps to integrate Eq. (4):

$$\mathbf{u}^{n+1} = L(\mathbf{u}^n, \mathbf{u}^n) \tag{5}$$

Here, we denote $\overline{\mathbf{u}} = L(-\mathbf{u}^n, L(\mathbf{u}^n, \mathbf{u}^n))$, $\mathbf{g}$ is the gravitational acceleration, and I is the identity matrix. An overview of our numerical method for solving these equations at each grid time step is given in Algorithm 1.

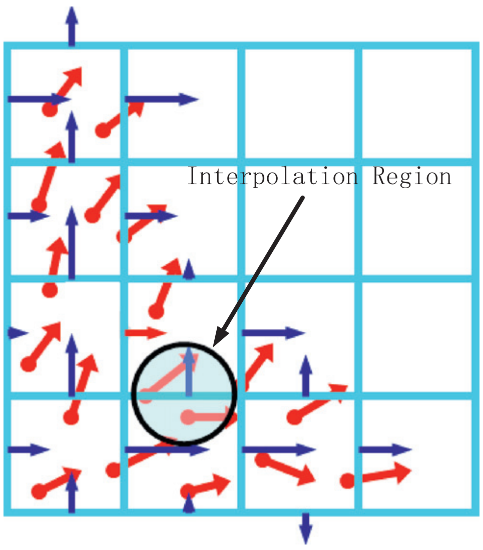

Fig. 6. The acceleration values are transferred to the background grid by using weighted averages. Red color represents the acceleration field of guiding particles, blue color represents the acceleration field of grid fluid. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Algorithm 1. Overview of fluid simulations on a grid.

```
1. BFECC advection:
   $\mathbf{u}^{n+1} = L(\mathbf{u}^n, \mathbf{u}^n + \frac{1}{2}(\mathbf{u}^n - \overline{\mathbf{u}}))$
2. Diffuse velocity:
   $(I - \upsilon \Delta t \nabla^2)\mathbf{u}^{n+1} = \mathbf{u}^n$
3. Add force:
   $\mathbf{u}^{n+1} = \mathbf{u}^n + \mathbf{g}\Delta t$
4. Project:
   $\nabla^2 p = \nabla \cdot \mathbf{u}$, then subtract $\nabla p$ from $\mathbf{u}$
5. Vorticity confinement:
   $\mathbf{u}_{grid} = \mathbf{u} + \mathbf{f}_{vort}\Delta t$
```

The computation of ink particle dynamics includes three stages: ink generation, ink movement, and ink extinction. An ink particle is

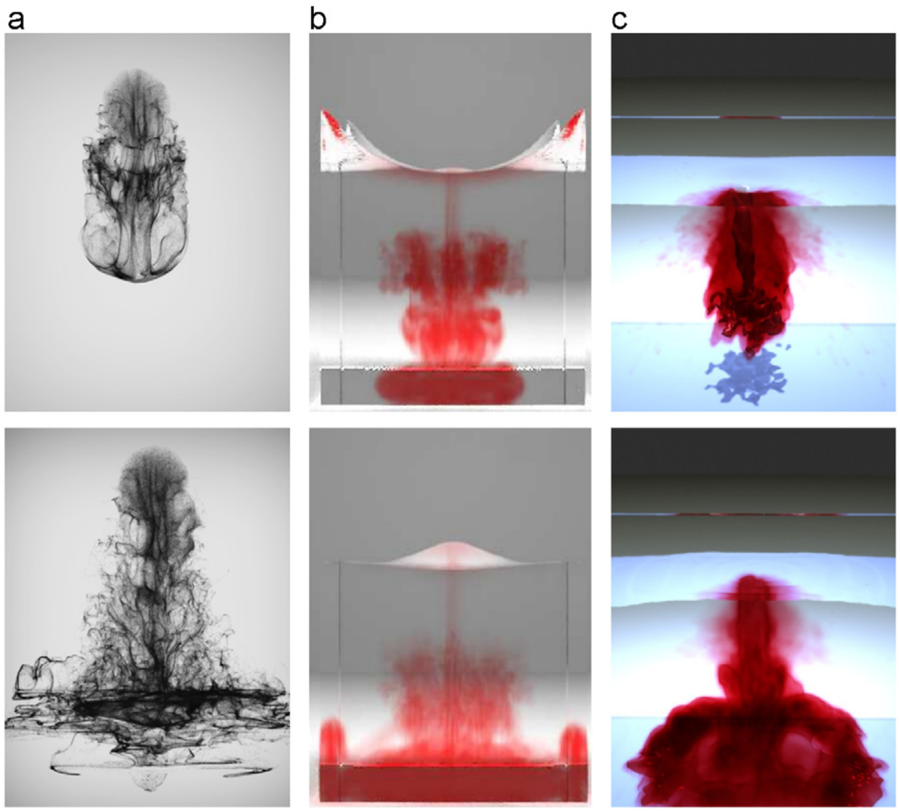

Fig. 7. The comparison between our method with other methods. (a) Ink diffusion using our method. (b) Ink diffusion using the method of Nahyup. (c) Ink diffusion using the method of Seung-Ho.

generated with an initial position and velocity, and then it will mix with the fluid flow. When an ink particle reaches its lifespan, we consider the ink particle to be extinct and set its transparency to one. We employ the following equations to calculate the advection of ink particles according to the Lagrangian method:

$$\mathbf{f}_{advection} \cdot \Delta t = \rho \cdot \mathbf{u}_{grid} \tag{6}$$

$$\mathbf{u}_{particle} = \mathbf{f}_{advection} \cdot \Delta t \tag{7}$$

where ρ is the fluid density in each grid cell, $\mathbf{u}_{grid}$ is the fluid velocity in each grid cell, $\mathbf{u}_{particle}$ is the velocity of each particle, and $\mathbf{f}_{advection}$ is the advection acceleration field of the ink particles. For each ink particle, we transfer the advection acceleration from the corresponding grid cell that contains water molecules using Eq. (6). The velocity of every particle is calculated using Eq. (7), which uses the grid variables to replace the velocity of particles according to the changes computed on the grid.

Then, we use the following equations to update the position of each ink particle:

$$\mathbf{Pos}^{n+1/2} = \mathbf{Pos}^{n} + \tfrac{1}{2}\Delta t' \cdot \mathbf{u}^{n} \tag{8}$$

$$\mathbf{Pos}^{n+1} = \mathbf{Pos}^{n+1/2} + \Delta t' \cdot \mathbf{u}^{n+1/2} \tag{9}$$

where **Pos** is the position of each particle, and **u** is the velocity of each particle. Once we have obtained the velocity field from the water grid cells, we can move the ink particles. Each particle's position is updated using Eqs. (8) and (9). The forward Euler method is sometimes adequate, but significantly better results can be obtained by using a slightly more sophisticated technique, such as a higher-order Runge–Kutta method. Here, we use the second-order Runge–Kutta method as our default because it is sufficiently accurate for our simulation. In particular, the particle time step is not required to be equal to the grid cell time step, so here we use $\Delta t'$ to represent the particle time step. However, the time steps must obey the CFL condition so that a particle moves less than one grid cell in each substep.

It is crucial that the value of the velocity outside the water region is zero, so that ink particles cannot flow out of the water region; this constraint also implies that the region of water grid constrains the simulation boundary for ink particles. However, PIC and FLIP cannot treat this situation because of the auxiliary character of their grids. Because the velocities in adjacency grid cells are different, particles in the same grid cell tend to generate thin filaments in the fluid flow. Importantly, the internal detail of the ink flow will still be preserved even when the grid is low-resolution. Fig. 7(a) shows the process of ink diffusing in water simulated using our method. As liquid ink is dropped into the water, the surface will spread out and become more transparent, which makes the internal fluid effects more obvious. We can produce the special effects of pinned boundaries and filament patterns. Because other regions of the water without ink particles are unimportant for practical animations, we simulate only the water flow mixed with ink particles.

We compare our method with other methods in Fig. 7. Fig. 7(b) and (c) shows the results of simulations performed using the methods of Nahyup [27] and Seung-Ho [2], respectively. We calculated approximately 30 frames per second for five million particles using our grid-particle representation. Compared to the methods of Nahyup [27] and Seung-Ho [2], which require multiple seconds to calculate each frame, our method is more efficient. Our method can better resolve the fine details of the ink flow and the complex fluid surfaces naturally. In reality, ink particles should spread out and become transparent as time passes, which is also required in animation production; otherwise we could not see the entire scene. Therefore, our simulation is realistic enough for art productions, and it can be controlled by the artist at high level and in real time.

4. Coupling with solids and other fluids

In the simulation, the solid–fluid interaction must satisfy no-penetration boundary conditions and pure Neumann pressure boundary conditions, which allow the fluid to slip tangentially past but not flow through the solid. The coupling between the fluid and the static container is a basic solid–fluid interaction, which assumes that the fluid domain is restricted by the container based on the simulation grid. Thus the normal component of the velocity and the normal pressure derivative are required to be zero at the boundaries:

$$\mathbf{u} \cdot \mathbf{n} = 0 \tag{10}$$

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \tag{11}$$

However, in practical animations, the fluid domain should be unrestrained and may change over time because of the presence of dynamic obstacles in the scenes. In general, we require the normal component of the fluid velocity to match the normal component of the solid's velocity such that the relative velocity has normal component zero:

$$\mathbf{u} \cdot \mathbf{n} = \mathbf{u}_{solid} \cdot \mathbf{n} \tag{12}$$

In practice, the velocities of dynamic obstacles are used to solve for the divergence and the vorticity in our simulation: we check whether the neighbor contains a solid obstacle, and if it does, we replace the value stored in the fluid's velocity field with the obstacle's velocity. We apply a similar method to solve for the pressure by replacing the pressure value for the center of the fluid cell with that of the neighbor solid cell.

Because we cannot always solve the Navier–Stokes equations to convergence, we explicitly enforce the no-penetration boundary condition immediately following the solution of the velocity. We correct the fluid's velocity component in the direction normal to the boundary with the corresponding component of the neighbor obstacle's velocity such that the normal relative velocity is zero. Fig. 8 shows the interaction of the ink and the dynamic obstacle; here, we approximate the obstacle as basic ball, which can be described with an implicit equation.

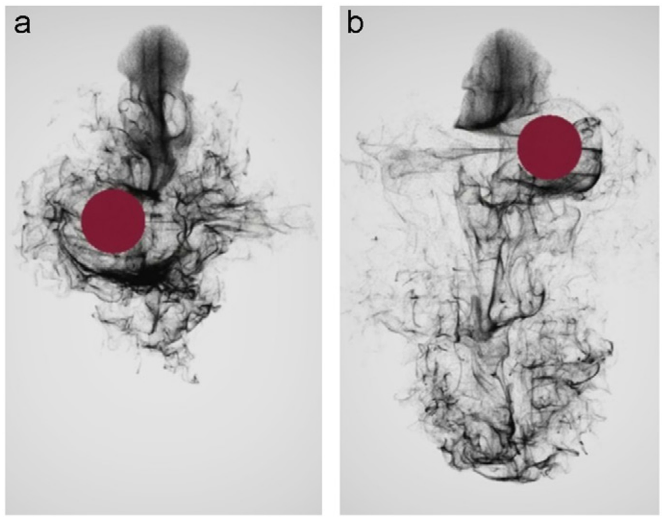

Fig. 8. The interaction of ink flow and dynamic obstacle. The ink–solid interaction satisfies no-penetration boundary conditions, the velocity of ink particles will change over time due to the dynamic solid.

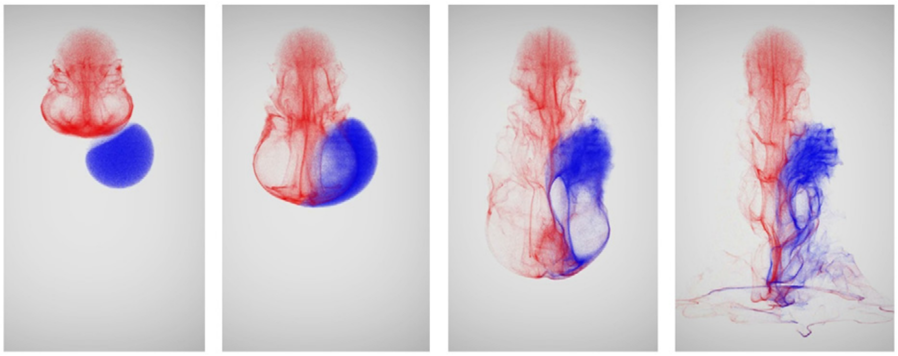

Fig. 9. The interaction of red ink and blue ink. Two kinds of ink particles are controlled by the same grid water, there is an implicit interaction between them. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

There is another type of coupling in which we are interested, which is the interaction of different types of inks. When inks are mixed, they spread out irregularly to form a fluid mixture with complex interface. To simulate ink mixing, not only the fluids but also the interface between them must be treated. However, it is difficult to directly simulate the interface between inks because of their complex geometry nature. Most of the existing methods that depend on the level-set method [6] cannot easily treat fluid interfaces naturally, and thus, they are not directly applicable to animation production.

Our improved grid-particle representation based on the components of liquid ink can naturally preserve the complex geometry nature of the interface of different inks with some modifications: one velocity field can drive a number of independent ink flows, each with its own physical properties; although each ink flow is controlled independently, there are implicit interaction conditions between them because they are advected by the same velocity field; and multiple velocity fields can drive a single ink flow. Given n ink particle flows $\mathbf{u}^1_{particle}, \ldots, \mathbf{u}^n_{particle}$ and m grid velocity fields $\mathbf{u}^1_{new}, \ldots, \mathbf{u}^m_{new}$, the only change in Eq. (6) is that the advection field is now the sum of the m grid velocity fields, and we now have n advection equations for the ink particle flows rather than one in Eq. (7):

$$\mathbf{f}_{advection} \cdot \Delta t = \sum_{i=1}^{m} \rho^i \cdot \mathbf{u}^i_{grid} \tag{13}$$

$$\mathbf{u}^j_{particle} = \mathbf{f}_{advection} \cdot \Delta t \quad (j = 1, \ldots, n) \tag{14}$$

Note, there can be one or more grid velocity fields in Eq. (13); the number depends on the m variable. Fig. 9 shows the situation where one grid velocity field drives two types of ink flows. We can see that the interface between the red ink and the blue ink is very realistic and natural.

5. Drag-force model for moving objects

In practical animations, water grid cells and ink particles can be emitted from various types of sources; therefore, we cannot ignore the possibility that not only the solids but also the sources are dynamic. For sources moving through a fluid, because of the water flow around the sources, ink particles emitted should face a resistive force from the fluid flow that is directed opposite to the direction of their movement; otherwise, the ink flow will be artificial. Therefore, we develop a drag field approach to extend our method.

In fluid dynamics, drag, or fluid resistance, refers to forces that oppose the motion of an object moving through a fluid.

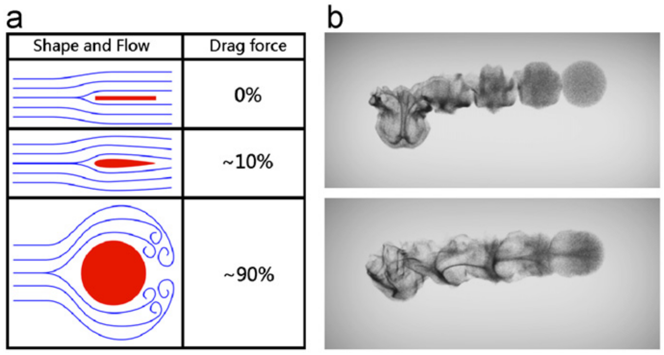

Fig. 10. The drag force model. The strength of drag force is affected by the shape of the objects. Drag force can improve look of rapidly moving emitters by eliminating the gaps between them. (a) Shape and drag force. (b) The effect of drag force.

Drag forces act in a direction opposite to the incoming flow velocity. Unlike other resistive forces, such as dry friction, which is nearly independent of velocity, drag forces depend on velocity. In reality, the strength of the drag force is affected by the shape of the objects, as shown in Fig. 10(a). The shape of ink particle is not considered here, and we use the shape of the emitter to directly compute the drag force on the ink particles.

The drag equation can be used to calculate the force using the relative velocity and the object shape as follows:

$$\mathbf{f}_{drag} = -\frac{1}{2}\rho A C_d (\mathbf{v} \cdot \mathbf{v}) \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{15}$$

where $\mathbf{f}_{drag}$ is the force vector of the drag, ρ is the density of the fluid, $\mathbf{v}$ is the velocity of the object relative to the fluid, A is the reference area, and C_d is the drag coefficient. The reference area A is often defined as the area of the object's orthographic projection on a plane oriented perpendicular to the direction of motion. The drag coefficient is a constant that we can use to control the drag force. We apply a drag force on ink particles when the objects move at high speeds. The drag force acting on a particle is calculated using Eq. (15). Therefore, we rewrite Eq. (7) with the drag force as follows:

$$\mathbf{u}_{particle} = \mathbf{f}_{advection} \cdot \Delta t + \mathbf{f}_{drag} \cdot \Delta t \tag{16}$$

There is a condition that we must consider: as the sources travel and time passes, an ink particle is located farther away from the sources, so the effect of the drag force must be reduced. Fig. 10(b) shows a comparison of simulation results before and after the drag force is included. The drag force can improve the appearance of rapidly moving objects by eliminating the gaps between them. In the upper panel of Fig. 10(b), we can observe that if the virtual object moves from left to right at a high speed and there is no drag force on the ink particles, the ink flow will appear discontinuous, which is not desirable for practical animations. From the lower panel of Fig. 10(b), we see that the results of the simulation including the drag force appear more realistic.

In summary, we modify the particle simulation steps and provide an improved version of the particle simulation, which is given in Algorithm 2.

Algorithm 2. Improved particle simulation.

```
1. Velocity transmission:
   $\mathbf{f}_{advection} \cdot \Delta t = \sum_{i=1}^{m} \rho^i \cdot \mathbf{u}^i_{grid}$
2. Drag force:
   $\mathbf{f}_{drag} = -\frac{1}{2}\rho A C_d (\mathbf{v} \cdot \mathbf{v}) \frac{\mathbf{v}}{\|\mathbf{v}\|}$
3. Update velocity:
   $\mathbf{u}_{particle} = \mathbf{f}_{advection} \cdot \Delta t + \mathbf{f}_{drag} \cdot \Delta t$
4. Update position:
   $\mathbf{Pos}^{n+1/2} = \mathbf{Pos}^n + \frac{1}{2}\Delta t' \cdot \mathbf{u}^n_{particle}$
   $\mathbf{Pos}^{n+1} = \mathbf{Pos}^{n+1/2} + \Delta t' \cdot \mathbf{u}^{n+1/2}_{particle}$
```

6. Motion blur-based rendering

We take advantage of the features of the GPU hardware, render the ink particles with texture mapping and alpha blending, and achieve more realistic results in real time. In our implementation, each ink particle is rendered as a particle sprite that is always window-aligned. Compared to other rendering methods, such as volumetric fluid rendering, particle sprite rendering is a better choice because it can efficiently yield good effects. However, there is a major problem with particle sprite rendering: it is very easy to obtain an undesirable grainy appearance. Instead of generating more particles, we employ a motion blur technique for particle rendering to smooth the final simulation and achieve better animation effects. To speed up the simulation, we save several positions along an ink particle's movement. Each ink particle has a head pointer, which is the current position, and a tail pointer, which is the previous position. In the implementation, we interpolate new particles between the current and previous particle positions, and their colors will fade in order.

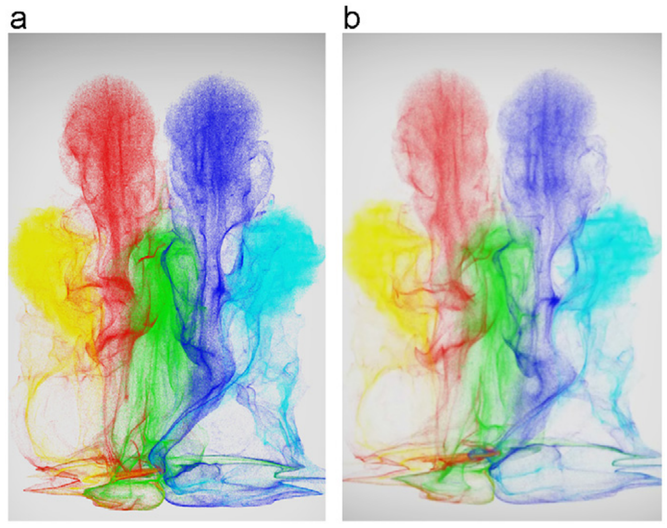

Fig. 11. The effect of motion blur technique. Motion blur can improve look of rapidly moving ink particles by eliminating the gaps between them. (a) Ink particles rendering without motion blur. (b) Ink particles rendering with motion blur.

Like the effect of the drag force, motion blur can improve the appearance of rapidly moving ink particles by eliminating the gaps between them, as Fig. 11(a) shows. The grainy appearance of the animation is very obvious when the motion blur is not performed, as shown in Fig. 11(b).

7. Implementation and results

We have implemented a standalone simulation system running on a GPU. We use uniform grids to discretize the simulation space, and the ink particle time step was set to 1.5 times the water grid cell time step. The most time-consuming part of the simulation is the solver for the water grid cells; therefore, we choose to implement our method on graphics hardware because such hardware is well suited to the type of computations required by grid fluid simulations. All the simulations were performed on an AMD PC with an NVIDIA GeForce GTX 460 graphics card. The computational speed for each of our simulations is given in Table 1. First, the frame rates drop rapidly when the particle number is six million; this decrease is caused by the performance of graphics card. Second, lower grid resolution and particle numbers lead to artificial simulation results. Finally, higher grid resolution and particle numbers require more computational resources and only yield minor improvements. Therefore, we typically use five million particles and a $50 \times 50 \times 50$ grid; these settings work best for obtaining the desired amount of fine detail. Next, We will review the examples of our method one by one.

In the first example, we simulate the process of ink diffusion in water in a container based on the grid. The ink particles are emitted from a virtual ball, as shown in Fig. 7(a), and the fluid surface spreads out and becomes more transparent, which makes the internal fluid effects more obvious. The simulation is sufficiently realistic for art production.

In the second example, we consider the interaction between ink particles and a dynamic obstacle, as shown in Fig. 8. The velocity of the ink particles changes over time because of the dynamic solid ball, and the fluid cannot flow through the solid because of the no-penetration boundary conditions and the pure Neumann pressure boundary conditions.

In the third example, we focus on an interaction of different types of ink flows. Fig. 9 shows the coupling between the red ink and the blue ink. Both the red ink and the blue ink are controlled by the same water grid cells, so there is an implicit interaction between them. We can see them spread out irregularly to form a fluid mixture with a complex interface.

In the fourth example, we add a drag force to treat the discontinuous phenomena caused by the emitters moving at high speed, as shown in Fig. 10(b). In the upper panel of Fig. 10(b), there is an object without a drag force. When the object moves from left to right, the look of the ink flow when the drag force is included, which is shown in the lower panel of Fig. 10(b), is more realistic and natural.

In the last example, we combine particle sprite rendering and the motion blur technique to improve the rendering results of the

Table 1
Performance analysis for different grid resolution and particle numbers. In our simulations, we typically use five million particles and a $50 \times 50 \times 50$ grid. Other resolutions and particle numbers can be used, but our experiments have shown that these settings are best for obtaining the desired amount of fine detail.

Grid	Four million (fps)	Five million (fps)	Six million (fps)
25^3	45	36	30
50^3	33	30	25
75^3	25	21	10

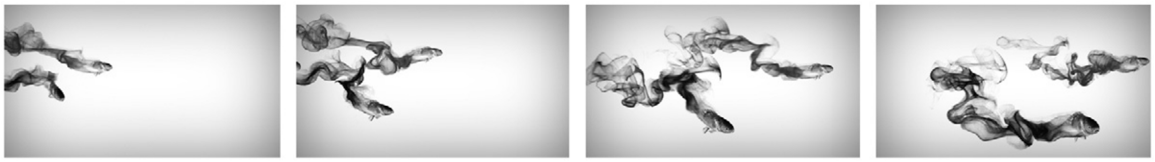

Fig. 12. An example of ink-made fish animation created with our method. Two fish are swimming in water freely, and the fishtails swing in high speed. Ink flows emit from the fishtails at the same time, then the attractive dynamic ink diffusion effects can be observed.

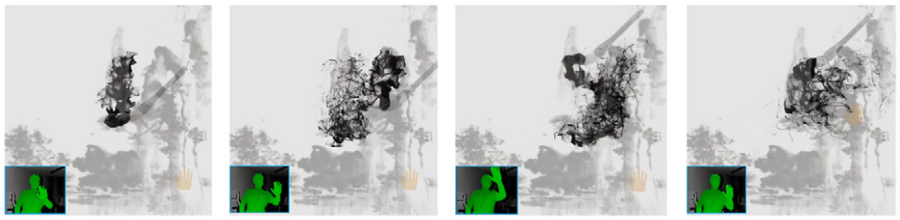

Fig. 13. The ink interaction system using Kinect. There is a virtual brush and a virtual hand, you can control them with the wave of a hand.

ink particles. Fig. 11 shows the interaction of five types of ink flows, where all the ink particles are advected by two water grid cells. In Fig. 11(a), ink particles are rendered in normal circumstances without motion blur; the grainy appearance of the animation is very obvious. In Fig. 11(b), motion blur is employed; it is useful to improve the appearance of rapidly moving ink particles by eliminating the gaps between them.

Our goal is to model the fine details of ink diffusion in practical animation and provide the maximum degree of control and flexibility to artists; we implement our grid-particle representation as a plug-in for SideFX Houdini and Autodesk Maya. We will show two examples of ink animation performed using our ink plug-in.

Fig. 5 shows an ink diffusion animation computed with our particle-guided method where the ink flow spreads outward in every direction. First, we use some coarse-resolution particles to simulate the large-scale motion. Actually, we apply the external force in different directions, which is not easy to implement in traditional grid-based method. Once we are satisfied with the large-scale motion of the ink flow, the acceleration field of the guiding particles is transferred to the water grid. Then, we can apply our grid-particle solver to simulate the ink flow.

Fig. 12 shows an ink fish animation created with our method in Houdini. In the animation, we use six million particles to simulate the fluid. Ink particles are emitted from the surface of fish. Because the fish swims in the water, we apply drag force model to the simulation to make the animation more realistic, and the motion blur technique is used to make the fluid smooth.

Currently, Microsoft Kinect is very popular. It provides an extraordinary way to interact with the world without using a controller. As an extended application, we have implemented an interaction system by combining our ink simulation with Kinect, as shown in Fig. 13. In the interaction system, there is a virtual brush and a virtual hand. Artists can control the brush to paint many special effects of ink flow with the wave of a hand, they can switch to control the virtual hand by pushing their hands, and they can feel the interaction between the ink and their hands.

8. Conclusions

We have developed a novel hybrid grid-particle method to simulate smaller-scale features, such as pinned boundaries and filament patterns that appear when ink spreads out in water, realistically in real time. To facilitate user interaction, we propose a particle-guided method to quickly preview the large-scale motion. To combine the attracting effects of ink diffusion with practical animation, we also introduce an extra drag force and motion blur technique to enhance the particle motion effects and make the animation realistic. Our method satisfies the requirements of practical industrial animation, it can address ink–ink coupling and ink–solid coupling, and it can generate the natural interfaces of different fluids without any additional calculations. We have implemented the simulation on graphics hardware in real time; thus, artists can easily interact with the simulation. Our method is well suited for performing realistic computer animations.

There are also some limitations of our method. First, motion blur-based rendering is not sufficient for large-scene animation, and the grainy look yielded by particles sprite rendering will be obvious. Thus, the rendering technique should be improved. Second, only a simple particle system is used to simulate the motion of ink particles, but particle–particle interaction should be considered to simulate filament effects accurately. Finally, we only simulate the diffusion effects of liquid ink in the beginning, and regions of water without ink particles are neglected, so our method only simulates the water flow mixed with ink particles. Incorporating the free surface effect in ink simulations is necessary because it is useful to create more realistic simulations of ink flow. We could solve the problems above to further enhance the visual appearance of ink diffusion.

In the future, we also aim to simulate more special fluid effects in practical animations, which requires simulating the fine details of small amounts of fluids accurately.

Acknowledgments

The authors would like to thank China Central Television (CCTV) for the reference video named Making of Ink. We would also thank Nahyup and Seung-Ho for the comparison with their works. This work is supported in part by National Natural Science Foundation of China (Nos. 60902078, 61172104, 60833007), in part by the Beijing Natural Science Foundation (Content-Aware Image Synthesis and Its Applications, No. 4112061), and in part by Changzhou Science and Art Combination Technology Research Center of Chinese Academy of Sciences.

References

[1] Chu NS-H, Tai C-L. Moxi: real-time ink dispersion in absorbent paper. ACM Trans Graph 2005;24:504–11.

[2] Shiny S-H, Kamz HR, Kimx C-H. Hybrid simulation of miscible mixing with viscous fingering. Comput Graph Forum 2010;29:675–83.
[3] Kunii T, Nosovskij G, Vecherinin Y. Two-dimensional diffusion model for diffuse ink painting. Int J Shape Modeling 2001;7:45–58.
[4] Swider JR, Hackley VA, Winter J. Characterization of Chinese ink in size and surface. J Cult Heritage 2003;4:175–86.
[5] Selle A, Rasmussen N, Fedkiw R. A vortex particle method for smoke water and explosions. ACM Trans Graph 2005;24:910–4.
[6] Enright D, Marschner S, Fedkiw R. Animation and rendering of complex water surfaces. ACM Trans Graph 2002;21:736–44.
[7] Nguyen DQ, Fedkiw R, Jensen HW. Physically based modeling and animation of fire. ACM Trans Graph 2002;21:721–8.
[8] 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, SCA '03. Aire-la-Ville, Switzerland: Eurographics Association; 2003. p. 154–9.
[9] Becker M, Teschner M. Weakly compressible sph for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA '07. Aire-la-Ville, Switzerland: Eurographics Association; 2007. p. 209–17.
[10] Clavet S, Beaudoin P, Poulin P. Particle-based viscoelastic fluid simulation. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA '05. New York, NY, USA: ACM; 2005. p. 219–28.
[11] Goswami P, Schlegel P, Solenthaler B, Pajarola R. Interactive sph simulation and rendering on the gpu. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10. Aire-la-Ville, Switzerland: Eurographics Association; 2010. p. 55–64.
[12] Lenaerts T, Dutré P. Unified sph model for fluid-shell simulations. In: ACM SIGGRAPH 2008 posters, SIGGRAPH '08. New York, NY, USA: ACM; 2008. p. 12:1.
[13] Kim J, Cha D, Chang B, Koo B, Ihm I. Practical animation of turbulent splashing water. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA '06. Aire-la-Ville, Switzerland: Eurographics Association; 2006. p. 335–44.
[14] Zhu Y, Bridson R. Animating sand as a fluid. ACM Trans Graph 2005;24: 965–72.
[15] Xu S, Mei X, Dong W, Zhang Z, Zhang X. Interactive visual simulation of dynamic ink diffusion effects. In: Proceedings of the 10th international conference on virtual reality continuum and its applications in industry, VRCAI '11. New York, NY, USA: ACM; 2011. p. 109–16.
[16] Foster N, Metaxas D. Realistic animation of liquids. Graph Models Image Process 1996;58:471–83.
[17] Stam J. Stable fluids. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques, SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1999. p. 121–8.
[18] Fedkiw R, Stam J, Jensen HW. Visual simulation of smoke. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques, SIGGRAPH '01. New York, NY, USA: ACM; 2001. p. 15–22.
[19] Angelidis A, Neyret F. Simulation of smoke based on vortex filament primitives. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA '05. New York, NY, USA: ACM; 2005. p. 87–96.
[20] WeiBsmann S, Pinkall U. Filament-based smoke with vortex shedding and variational reconnection. In: ACM SIGGRAPH 2010 papers, SIGGRAPH '10. New York, NY, USA: ACM; 2010. p. 115:1–12.
[21] Kim B, Liu Y, Llamas I, Rossignac J. Advections with significantly reduced dissipation and diffusion. IEEE Trans Vis Comput Graph 2007;13:135–44.
[22] Harlow FH. The particle-in-cell method for numerical solution of problems in fluid dynamics. In: Experimental arithmetic, high-speed computations and mathematics, 1963.
[23] Brackbill J, Ruppel H. Flip: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensionsJ Comput Phys 1986;65:314–43.
[24] Zhu H, Liu X, Liu Y, Wu E. Simulation of miscible binary mixtures based on lattice Boltzmann method. J Vis Comput Animation 2006;17:403–10.
[25] Zhu H, Bao K, Wu E, Liu X. Stable and efficient miscible liquid–liquid interactions. In: Proceedings of the 2007 ACM symposium on virtual reality software and technology, VRST '07. New York, NY, USA: ACM; 2007. p. 55–64.
[26] Bao K, Wu X, Zhang H, Wu E. Volume fraction based miscible and immiscible fluid animation. Comput Animation Virtual Worlds 2010;21:401–10.
[27] Kang N, Park J, Noh J, Shin SY. A hybrid approach to multiple fluid simulation using volume fractions. Comput Graph Forum 2010;29:685–94.
[28] Guo Q, Kunii TL. Modeling the diffuse paintings of sumie. In: Kunii TL, editor. Modeling in computer graphics. Proceedings of the IFIP WG 5.10 working conference, Tokyo, April 1991. IFIP series on computer graphics. Tokyo, Berlin, Heidelberg: Springer-Verlag; 1991. p. 329–38.
[29] Zhang Q, Sato Y, Takahashi Jy, Muraoka K, Chiba N. Simple cellular automaton-based simulation of ink behaviour and its application to suibokuga-like 3d rendering of trees. J Vis Comput Animation 1999;10: 27–37.