DOI: 10.1111/cgf.12488
Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Hybrid Particle-grid Modeling for Multi-scale Droplet/Spray Simulation

Lipeng Yang[1], Shuai Li[1*], Aimin Hao[1], and Hong Qin[2]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. Corresponding Author: Shuai Li, ls@vrlab.buaa.edu.cn
[2] Stony Brook University, Stony Brook, USA, qin@cs.stonybrook.edu

Abstract

This paper presents a novel hybrid particle-grid method that tightly couples Lagrangian particle approach with Eulerian grid approach to simulate multi-scale diffuse materials varying from disperse droplets to dissipating spray and their natural mixture and transition, originated from a violent (high-speed) liquid stream. Despite the fact that Lagrangian particles are widely employed for representing individual droplets and Eulerian grid-based method is ideal for volumetric spray modeling, using either one alone has encountered tremendous difficulties when effectively simulating droplet/spray mixture phenomena with high fidelity. To ameliorate, we propose a new hybrid model to tackle such challenges with many novel technical elements. At the geometric level, we employ the particle and density field to represent droplet and spray respectively, modeling their creation from liquid as well as their seamless transition. At the physical level, we introduce a drag force model to couple droplets and spray, and specifically, we employ Eulerian method to model the interaction among droplets and marry it with the widely-used Lagrangian model. Moreover, we implement our entire hybrid model on CUDA to guarantee the interactive performance for high-effective physics-based graphics applications. The comprehensive experiments have shown that our hybrid approach takes advantages of both particle and grid methods, with convincing graphics effects for disperse droplets and spray simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction and Motivation

In the most recent years many researchers are starting to focus on the fluid detail enhancement that is neglected in commonly-used macro-scale fluid solvers. For example, fluid details and their natural interaction with nearby liquids must be vividly simulated when exhibiting natural phenomena with certain dramatic visual effects, such as high-pressure water stream, fountain, waterfall, and smash waves, wherein varying-scale fluid particles are emitted from liquid stream ranging from apparent droplets to minute droplets and visible fog-like sprays. Of which, one of the most important challenges is how to interactively simulate droplets, fog-like spray and their natural inter-transition.

Among state-of-the-art fluid simulation methods, Lagrangian particles are widely used to represent individual droplets [IAAT12]. In particular, they are also popularly employed to capture the missing sub-grid details [TFK*03, KCC*06, MMS09] in Eulerian fluid methods, because the apparent dispersing droplets can be naturally represented and rendered with spheres/ballistic particles. Meanwhile, refining the particle scales could facilitate the visual effect enhancement, yet it is at the expense of much heavier computational cost, especially when inter-collision of particles needs to be synchronously handled in more accurate simulation. Thus, Lagrangian particle based methods are hard to simulate sprays comprising large amounts of minute droplets. As for volumetric spray, since the shape of droplet is not visually obvious [NØ13], it usually leverages Eulerian based methods instead. By solving incompressible Navier-Stokes equations on an auxiliary grid, Eulerian density field can be advected by a divergence-free velocity field. Although the Eulerian model gives rise to the flexible transport, diffusion and dissipation of spray, it is hard to explicitly represent apparent droplets emitted from the massive liquid body nearby, and fails to exhibit the vivid droplet-liquid interaction as well

as the natural inter-transition among droplets, spray, and liquid stream.

Strongly motivated by the urgent need for visual-detail enhancement of fluid simulation, this paper aims to simultaneously simulate individual droplets and fog-like sprays while accommodating physical correctness and high-effective performance. Therefore, we propose a novel particle-grid hybrid solution to tackle multi-scale fluid modeling (e.g., water, droplets, and spray) as well as their interactive and realistic simulation. Our hybrid model can fully leverage the respective advantages of Lagrangian and Eulerian methods, wherein particles are employed for individual droplet representation and the grid-based density field is introduced for volumetric spray modeling. Moreover, this new hybrid model is built upon the smart and effective intertwinement of the particle system and the grid density field, wherein the size distributions and transitions of droplets are handled at the geometric level while the dynamics of two models are tightly coupled at the physical level. Specifically, the salient contributions of this paper include:

- We propose a novel hybrid Lagrangian-Eulerian framework to model the behaviors and their natural mixtures of multi-scale droplets and fog-like spray, which can enable physics-meaningful fluid-detail enhancement via mass-conserving smooth transition among liquid, droplet, and fog-like spray.
- We formulate a new FLIP-Lagrangian model to simulate the vivid interaction among liquid, droplet, and spray by introducing the rigorous droplet-spray drag force definition and employing the liquid velocity field as the boundary condition of spray velocity field.
- We design GPU-based algorithms to facilitate the parallel implementation of the physical simulation, transition, and interaction, which collectively guarantee the interactive performance of our hybrid model.

2. Background and Related Work

Many visual-effect-dominant fluid simulation methods have been introduced in computer graphics, including Smoothed Particle Hydrodynamics (SPH) [Mon92, MCG03, IOS*14], Level Set method [EMF02, FF01], Fluid Implicit Particle (FLIP) method [ZB05], etc. Among them, FLIP method avoids numerical dissipation while achieving stable incompressibility by way of combining Lagrangian particles and grid-based Poisson solver. Recently, FLIP method has been further extended to simulate multi-phase fluid [BB12], preserve fluid sheet [ATT12], model splashing water [GB13], combine with SPH [CIPT14], etc. We employ FLIP model mainly because it utilizes particles to represent fluid, whose mass conservation can be straightforwardly enforced while re-sampling can also be avoided by employing the improved FLIP model [ATT12]. Meanwhile, by resorting to the employed Eulerian Poisson solver, it avoids the stability problem existed in SPH-like particle based methods, wherein various sizes of particles are employed and particle split and merge could be dynamically accommodated. Besides, to reduce computational cost while maintaining the smoothness of fluid surface, adaptive sampling strategy is introduced into particle-based fluid simulation [APKG07, YWH*09, SG11], later it is also employed in FLIP model [ATT12]. Relevant to the central theme of this paper, we now briefly review the previous works related to droplet/spray simulation.

Particle-based droplet/spray. To enhance the details of free-surface fluid simulation, particles are mostly used to represent the individual sub-grid droplets escaping from Eulerian based liquid body. [TFK*03] proposes to indicate such disperse droplets by generating massless particles in high curvature regions. To involve the mass transfer in droplet generation, [KCC*06] further computes the volume loss of Particle Level Set fluid, and then redistributes the lost mass to generate droplet particles. And the droplet is absorbed when it hits the level set interface, while its mass and momentum is added into the main body of fluid. To improve the quality of bubble/droplet generation, [MMS09] introduces Weber number as a physical criterion to generate secondary particles, wherein a drag force model for the interaction of droplets and the surrounding air is also presented. Meanwhile, [TRS06, CM10] utilize a similar particle system to capture the lost details (disperse droplets, spray and foam) in shallow water solver. Instead of using a simple particle system, [LTKF08] employs SPH to simulate disperse droplets produced from level set fluid. [IAAT12] proposes a unified particle-based model to enhance the results of a SPH solver by simulating diffuse materials (spray, foam and bubble) via post-processing.

However, particle based models have difficulties to simulate spray and mist that are quite light and will dissipate in air gradually. Increasing particle number is a simple way but tends to result in heavy computational cost, which makes the method not suitable for realtime computer graphics. Moreover, the above-documented methods (except for [KCC*06]) ignore the mass transfer between main body of water and droplets. In sharp contrast, we consider their two-way transition by enforcing the constraints of mass and momentum conservation, which is critical when the fluid volume loss is obvious.

Grid-based droplet/spray. Grid-based density field is widely used to model the minutely-dispersed droplets, such as cumulus, spray, etc. [MDN02] animates cumulus by solving the partial differential equation of atmospheric fluid model and advecting the density of water droplets accordingly. [MDCN03] argues that cloud can not be treated as fluid and then employs a two-fluid model to simulate volcanic clouds. [LWGP08] simulates the atmospheric binary mixture such as tornado by employing a two-fluid model, in which air flow and dust are simulated separately and their interaction is achieved by introducing an extra interaction force. Most recently, [NØ13] employs a two-continua ap-

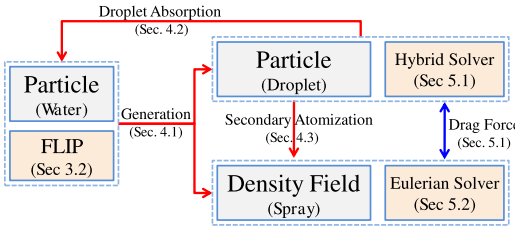

Figure 1: *The framework of our model.*

proach to get more physically plausible results by evolving velocity fields both for spray and the surrounding air. However, grid-based density field only works when the simulated droplets are minute and the individual droplet is not apparent. And the Eulerian solver tends to enforce incompressibility on density field, which is not suitable for heavy droplets that have ballistic movement.

Hybrid particle-grid based droplet/spray. Besides the particle-based droplet model, [TFK*03, KCC*06, LTKF08] introduce an extra density field to represent the mist emitted by droplets, and enhance the rendering results. Although these works combine particle and density field together, they commonly ignore the physical transition process, wherein spray should be emitted from droplets with proper mass and momentum transfer. For turbulent water scenarios in movies, [GLR*06, FGP07] model both secondary droplets and spray with a hybrid model, but their post-processing models lack two-way droplet-liquid interaction and mass transport. To further improve, we shall focus on the physics-based droplet transition process, which is critical for violent water stream especially when the transition dominates the visual effects.

GPU-based implementation. In fluid simulation, more and more researchers seek to design GPU-based algorithms to improve the performance. For particle-based fluid simulation, [HKK07, GSSP10, IABT11, KE12] respectively propose a host of techniques to facilitate parallel computation, including neighborhood searching, rendering, etc. On the other hand, [Har04, CCLW11] implement grid-based fluid methods on GPUs to improve computational performance. In this paper, we design parallel computation algorithms for our hybrid particle-grid method, and we detail our GPU implementation in Section 6.2.

3. Method Overview and FLIP Review

3.1. Method Overview

As shown in Fig. 1, our hybrid model comprises three schematic parts that are tightly integrated: particle based water, particle based droplet, and grid based spray. From the view point of simulation and representation, adaptive FLIP is used to simulate the volumetric water, and FLIP is briefly introduced in Section 3.2; Lagrangian particles are used to represent disperse droplets, and a hybrid particle-grid solver is employed to drive them; while Eulerian based representation and solver are employed for spray simulation.

As indicated with the red arrows in Fig. 1, there are three types of transitions involved in our model: first, droplets and spray are generated from liquid particles where the flow of liquid stream is violent, described in Section 4.1; second, droplets entering the volumetric water will be absorbed in a mass-conserving way, wherein the momentum transference is also considered (Section 4.2). Third, we capture the secondary atomization phenomenon in which droplets crack to minute droplets, which is represented by grid based spray and detailed in Section 4.3.

As for interactions, when droplets are generated, the droplet-water interaction is reduced to droplet abortion; since the density of spray is ignorable compared with liquid, they are coupled by employing liquid velocity field as the boundary condition of spray velocity field; while the interaction between the droplets and spray are modeled via a drag force, which is detailed in Section 5.

3.2. FLIP Review

Our hybrid model is based on adaptive FLIP model [ATT12]. We now briefly review the basic idea of fluid simulation, FLIP model, and its possible improvement. Fluid dynamics is essentially based on Navier-Stokes equations (N-S equations) that conserve both mass and momentum:

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{2}$$

where ρ is the density, $\mathbf{u}$ is the velocity, p is the pressure and $\mathbf{f}$ is the external force.

In FLIP model, fluid is discretized as particles and traditional Eulerian method is employed to solve the N-S equations. Unlike Particle-in-Cell (PIC) method, the velocity change on grid, rather than the velocity itself, is interpolated over particles. As a result, the numerical dissipation problem is avoided, making FLIP more suitable for violent fluid simulation.

To alleviate the noisy-like behaviors of FLIP, similar to [ZB05], we linearly blend the PIC and FLIP velocities as:

$$\mathbf{v} = a\mathbf{v}_{FLIP} + (1-a)\mathbf{v}_{PIC}, \tag{3}$$

where the blend factor a is set to be 0.95 in all our experiments.

Besides, original FLIP method suffers from the uneven distribution of particles, which is often solved via re-sampling. To improve, [ATT12] proposes a position correction step, which is similar to the pressure step for SPH. Since re-sampling is avoided in [ATT12], the particle number is kept unchanged in the simulation. Thus it can easily make

FLIP retain the SPH-like mass-conserving feature. It may be noted that, the mass conservation is critical to our hybrid model together with the stability of FLIP for accommodating varying-scale particles.

4. Transition Models among Liquid, Droplet, and Spray

4.1. Droplets/Spray Generation from Liquid

For disperse droplets and spray, we specialize a generation step to model the crack of violent liquid stream. We first mark the liquid particles in FLIP model that meet the proposed criterion, then split the liquid particles to droplet particles, and emit spray at the same time. Although liquid and droplet are simulated with different solvers, directly benefiting from the underlying FLIP solver, the uneven distribution of particles due to particle transition and splitting will not cause the stability problem [ATT12].

4.1.1. Generation Criteria

Liquid streams tend to be unstable when interacting with the gaseous environment due to complex reasons, as described in [MMS09], wherein an important tradeoff is to introduce the Weber number:

$$We = \frac{\rho_a |\Delta \mathbf{v}|^2 L}{\sigma}. \tag{4}$$

Here ρ_a is the air density, $\Delta\mathbf{v}$ is the relative air-liquid velocity, L is the characteristic parameter of the liquid stream, and σ is the surface tension coefficient. According to the simplification scheme in [MMS09], we define $\gamma = |k||\Delta\mathbf{v}|^2$ as one criterion for liquid particle, where k is the surface curvature at the particle's position. γ can well measure the boundary condition between liquid stream and the surrounding air in the macro view. It still needs to measure the violent extent of the liquid stream itself. We conduct this task by considering the relative velocity $\mathbf{v}_i^{rel}$ between each particle and its surrounding particles:

$$\mathbf{v}_i^{rel} = \mathbf{v}_i - \frac{\sum_{j\in S_i} \mathbf{v}_j m_j W_{smooth}(d_{ij}, h)}{\sum_{j\in S_i} m_j W_{smooth}(d_{ij}, h)}. \tag{5}$$

Here S_i denotes the neighboring particle set of i-th particle. h is the kernel radius, which is relevant to the length of Eulerian grid cell. d_{ij} is the distance between i-th and j-th particles, and the weighting kernel $W_{smooth}(d,h)$ is the same as that in [ATT12].

This transition only occurs to the liquid particles near the interface of liquid and air, i.e., particles in the liquid surface cells, which can be easily found by dilating air cells a few times. After computing the above two criteria for each liquid particle, we mark the particle whose criteria value (γ or $\mathbf{v}_i^{rel}$) exceeds the corresponding threshold (γ_{th} or $\mathbf{v}_{th}^{rel}$), and transit it to droplets.

4.1.2. Droplet Particle Size Distribution

The droplet size has important influence on its behavior [Sir10]. Consider the fact that we only pay attention to the droplet's behavior modeling rather than its interior dynamics, we choose to ignore droplet deformation and use sphere to represent its shape. Thus, the droplet size can be determined by its diameter. Generally speaking, there are three ways to predict the droplet sizes, including empirical correlation, instability analysis of liquid stream, and maximum entropy principle [Sir10]. Here we employ the widely used Rosin-Rammler distribution equation (Eq. 6) to determine the droplet diameter x and its corresponding probability density function:

$$f(x) = \frac{b}{d_{aver}^b} x^{b-4} exp[-(\frac{x}{d_{aver}})^b]. \tag{6}$$

Here b is the spread parameter and d_{aver} is the average diameter of droplets. In all of our experiments, b is set to be 5.0, while $d_{aver} = 0.6 * d_l$ (d_l is the initial diameter of liquid particle, which is calculated according to the particle distribution while initializing the entire simulation). In practice, we discretize Eq. 6 by taking n diameters (denoted by d_i) as candidates, where n is a user-defined positive integer ($n = 6$ in all of our experiments). We make droplet diameters distribute uniformly in the diameter range $(0, d_l)$ by defining $d_i = \frac{i}{n+1} \cdot d_l$. Then we compute the corresponding probability f_i and get its normalized version as $f_i \leftarrow \frac{f_i}{\sum_{j=1}^n f_j}$.

For each liquid particle satisfying the criteria, next we need to transit it (diameter d_0, mass m_0) to droplet particles and spray. We transit a part of the liquid particle mass (cm_0) to spray directly, and the remaining mass ($(1-c)m_0$) is transited to droplets. For droplet generation, we iteratively pick a droplet diameter d_i randomly with probability f_i until the generated droplets consume all the mass $(1-c)m_0$. Meanwhile, we use n_{max} to limit the maximum droplets that a liquid particle can split into. After determining the diameters of newly generated droplets, their positions are sampled randomly within the space where the original liquid particle occupies, and the droplets inherit the velocity to conserve momentum. The implementation details of the parallel splitting algorithm are shown in Section 6.2.

4.2. Droplet Absorption to Liquid

When a droplet particle hits the liquid surface, it should be absorbed by the main body of liquid. Benefitting from that both droplet and liquid are represented by particles, we can dynamically change its types in runtime and subsequently use FLIP solver to govern its dynamics. However, the relative velocity between the droplet and its surrounding liquid particles should be large, which may cause noisy behavior of FLIP solver, so we model the collision and momentum exchange explicitly as follows.

Assume a droplet particle p_i hits liquid surface, we use S_i

to denote the surrounding particle set. According to the law of momentum conservation, the new velocity of p_i is:

$$\mathbf{v}^* = \frac{m_i\mathbf{v}_i + \sum_{j\in S_i} m_j\mathbf{v}_j}{m_i + \sum_{j\in S_i} m_j}, \tag{7}$$

where m and $\mathbf{v}$ denote the particles' mass and velocity. And for each surrounding liquid particle p_j ($j \in S_i$), we add the velocity increment $\mathbf{v}^{inc}$ (Eq. 8) to $\mathbf{v}_j$, which will not smooth the velocities of surrounding particles negatively.

$$\mathbf{v}^{inc} = \mathbf{v}^* - \frac{\sum_{j\in S_i} m_j\mathbf{v}_j}{\sum_{j\in S_i} m_j}. \tag{8}$$

When the transition process ends, small droplet particles will be added into FLIP solver. It should be noted that naive transition can increase the computational cost of FLIP solver due to introducing a large amount of small particles. To alleviate this problem, we further merge the small particles with an scheme similar to [ATT12]. And we improve it in the aspect of parallel implementation, which is detailed in Section 6.2.

4.3. Secondary Atomization of Droplets

When the relative droplet-air velocity is large enough, droplet will deform and fragment due to drag force, which is secondary atomization. We employ density field to represent minute droplets, and we transfer the mass from droplet particles to density field when secondary atomization occurs. To control the transition process, we set two thresholds v_{max}, v_{min} for the relative droplet-air velocity $\Delta\mathbf{v}$. We then can get a proportional value α, and transfer mass αm_i from droplets to spray, wherein m_i denotes the mass of the droplet particle p_i.

$$\alpha = \begin{cases} \beta & |\Delta\mathbf{v}| < v_{min} \\ \beta\frac{v_{max}-|\Delta\mathbf{v}|}{v_{max}-v_{min}} & otherwise \\ 0 & |\Delta\mathbf{v}| > v_{max} \end{cases} \tag{9}$$

Here β is a scale factor, which controls the transition degree from droplets to spray together with v_{min}, v_{max}. When the droplet particle is too small, we directly transfer all of its mass m_i to spray and remove the particle in droplet solver. In this model, apparent droplets with lower velocity transit to spray more likely. And Fig. 7 shows the effects of the parameters v_{min}, v_{max}, β.

Suppose that the mass αm_i of droplet p_i is transferred to the spray, i.e., to the neighboring grid cells within a transferring radius h. The corresponding mass change of a neighboring grid cell is computed as follows.

$$\Delta m_j = \frac{W_{smooth}(d_{ij},h)}{\sum_{j\in S_i} W_{smooth}(d_{ij},h)}\alpha m_i, \tag{10}$$

where $W_{smooth}(d,h)$ is the same kernel used in Eq. 5. And the momentum change of the neighboring grid cell is $\Delta m_j\mathbf{v_i}$, which gives rise to the update of grid cell velocity as:

$$\mathbf{v}_g^* = \frac{m_g\mathbf{v}_g + \Delta m_j\mathbf{v}_i}{m_g + \Delta m_j}, \tag{11}$$

where m_g and $\mathbf{v}_g$ are respectively the mass and velocity of the density field.

5. Hybrid Model for Droplet and Spray Simulation

5.1. Lagrangian-Eulerian Model for Droplet Simulation

We model the inter-collision by treating the droplets as liquid, and adopt FLIP solver to compute each droplet's velocity change, which in fact indicates the influence of surrounding particles. This Eulerian model treats droplets as an incompressible liquid, which is not suitable when droplets are sparse in space. So we complementarily introduce a Lagrangian model, in which the interactions among droplets are ignored. By uniting the Lagrangian and Eulerian models together via a density based coefficient, we can model the entire behavior range of the droplet varying from completely incompressible to ballistic. Based on the FLIP model introduced in Section 3.2, we detail our Lagrangian model and the coupling method as follows.

Lagrangian Model. There are three types of forces that may affect droplet: drag force from the surrounding air and spray, inter-droplets collision force and external forces such as gravity force. In practice, we ignore the inter-collision among droplets, and the external forces can easily be obtained in a routine way. Thus, the governing equation for individual droplet particle is defined as follows:

$$\frac{d\mathbf{v}_d}{dt} = \frac{\mathbf{f}_{drag} + \mathbf{f}_{ext}}{m_d}, \tag{12}$$

where $\mathbf{v}_d$ denotes the velocity of droplet particle, $\mathbf{f}_{drag}$ is the drag force, $\mathbf{f}_{ext}$ is the external force, and m_d is the mass. Meanwhile, we employ the model introduced in [MMS09, NØ13] to define the drag force:

$$\mathbf{f}_{drag} = \frac{1}{2}C_D(\rho_{sp} + \rho_{air})\pi r^2|\Delta\mathbf{v}|\Delta\mathbf{v}. \tag{13}$$

Here $C_D = 24/Re^{\theta}$ and θ is interpolated from 1 to 0.72 according to the reference Reynolds number interval [1,30]. ρ_{sp} is the spray density defined over grid, ρ_{air} is the air density, and $\Delta\mathbf{v}$ denotes the relative droplet-air velocity. To conserve momentum between spray and droplets, the drag force is also enforced for spray as documented in Eq. 17.

Coupling of Lagrangian and Eulerian Models. The interactions among droplets are heavily dependent on the average spacing, and the droplet is more likely to be affected by other droplets when the density at its position is high. We compute the density of droplet particle i as:

$$\rho_i = \sum_{j\in S_i} m_j W_{poly6}(d_{ij},h). \tag{14}$$

Here the kernel function $W_{poly6}(d,h)$ is from [MCG03]. Subsequently, we can get the tuning coefficient s_i of particle i through

$$s_i = \frac{\rho_i - \rho_{min}}{\rho_{max} - \rho_{min}}. \tag{15}$$

And ρ_{min} and ρ_{max} are a pair of thresholds that balance the affects of Lagrangian and Eulerian models, and s_i is confined within the interval $[0,1]$.

After we obtain the velocity $\mathbf{v}_L^{n+1}$ updated by Lagrangian method (Eq. 12) and $\mathbf{v}_E^{n+1}$ updated by Eulerian method (Eq. 3) respectively, we compute the velocity of droplet particle using

$$\mathbf{v}^{n+1} = (1-s)\mathbf{v}_L^{n+1} + s\mathbf{v}_E^{n+1}. \tag{16}$$

In our hybrid model, when the density ρ_i is low, the droplet will be transported in a ballistic way. As ρ_i increases, the dynamics of the droplet will be smoothly transferred to an incompressible mode, which provides us with a flexible scheme to tune droplet simulation via controlling ρ_{max} and ρ_{min}. Fig. 2 shows the effects of this coupling model: as the droplet density decreases, the affect of the inter-collision becomes weaker. For the clarity purpose, we detail the computation steps of our model in Algorithm 1.

Algorithm 1: Lagrangian-Eulerian Model for Droplet Simulation

1 $\mathbf{u}_{grid}^n \leftarrow \mathbf{v}^n$: map the particles' velocity to the grid.
2 $\mathbf{u}_{grid}^{n+1}$ = Project($\mathbf{u}_{grid}^n$): project the velocity field for incompressibility.
3 Update the result from Eulerian model $\mathbf{v}_E^{n+1}$ by Eq. 3.
4 Compute the drag force $\mathbf{f}_{drag}$ with Eq. 13.
5 Update the result for Lagrangian part $\mathbf{v}_L^{n+1}$ with Eq. 12.
6 Compute ρ_i for spray particles by Eq. 14.
7 Compute the tuning coefficient s_i by Eq. 15.
8 Update $\mathbf{v}^{n+1}$ according to Eq. 16.

5.2. Spray Simulation

Since spray comprises large amounts of very small droplets which mainly move along with air, we ignore the gravity and buoyancy, and utilize an incompressible solver to simulate it. The velocity field of spray $\mathbf{u}_{sp}$ is governed by

$$\rho\left(\frac{\partial \mathbf{u}_{sp}}{\partial t} + \mathbf{u}_{sp}\cdot\nabla\mathbf{u}_{sp}\right) = -\nabla p + \mathbf{f}_{drag}, \tag{17}$$

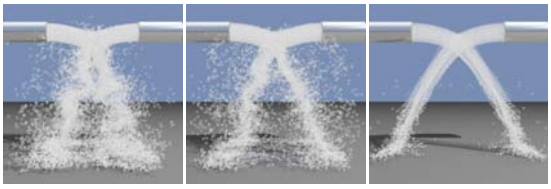

Figure 2: *Droplets. Droplets are emitted directly with different initial particle density. From left to right, the average density ρ of droplets before the collision of two streams are 1000, 500, and 200.*

which is solved by the FLIP solver.

For the density field of spray, we model the dissipation, diffusion, and the source item S representing spray generation as

$$\frac{\partial \rho}{\partial t} + \mathbf{u}_{sp}\cdot\nabla\rho = \mu\nabla^2\rho - d\rho + S, \tag{18}$$

where μ and d denote the diffusion and dissipation coefficients respectively. The density field is advected by a semi-Lagrangian scheme.

6. CUDA-based Implementation

6.1. Pipeline

For the purpose of reproduction, we detail the entire pipeline of our model in each simulation cycle as follows.

1. FLIP solver for liquid.
   - Project velocity change from grid to liquid particles, and update their velocities (Eq. 3) and positions subsequently.
   - Project velocity from particles to grid.
   - Make the velocity field divergence-free by solving the Poisson equation.
2. Droplet/spray simulation
   - Simulate droplets using the coupled Lagrangian-Eulerian model shown in Algorithm 1.
   - Solve N-S equations to obtain spray velocity field by taking drag force $\mathbf{f}_{drag}$ (Eq. 13) as an external force.
   - Advect the spray density field via Eq. 18.
   - Advect the spray velocity field via a semi-Lagrangian scheme [Sta99].
3. Transitions
   - For each liquid particle, compute the criteria γ and $\mathbf{v}_{rel}$, then generate droplet particles and obtain the spray density source (S in Eq. 18).
   - For each droplet particle, conduct absorbtion computation if it enters the main body of liquid.
   - For each droplet particle, transit it to spray according to the criterion in Section 4.3.
   - For each liquid particle, check and merge small liquid particles.

6.2. CUDA-based Numerical Computation

We implement the entire proposed method on CUDA for efficiency. While there have been other researchers designing parallel framework for both Eulerian and Lagrangian solvers, we focus on the parallel scheme for FLIP solver and droplet/spray simulation. Then we detail our CUDA-based data structure and computational scheme for the transitions among liquid, droplet and spray.

FLIP Solver for Liquid. For each particle, we invoke a

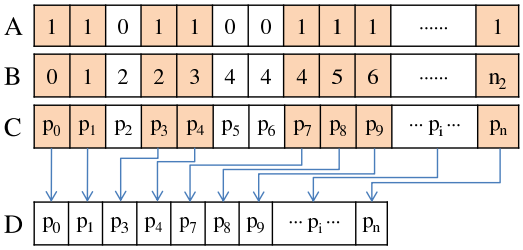

Figure 3: *CUDA-based memory compaction scheme.*

CUDA thread and calculate which grid cell it belongs to, then interpolate the velocity change from grid, subsequently we enforce the external force and update its velocity and position. After that, we invoke a CUDA thread for each grid cell and interpolate its velocity from particles. For each grid cell, we store the indices of particles that it contains in order to accelerate the interpolation. At last, we parallelize the conjugate gradient (CG) method to solve Poisson equation. For more details, please refer to our supplementary source codes.

Droplet/spray Simulation. For droplet simulation, the Lagrangian part is straightforward for parallel computation because each particle is not related to other particles, and the above FLIP solver is employed for the Eulerian part. To calculate the particle density ρ_i for the coupling parameter s_i (Eq. 15), we invoke a CUDA thread for each particle and search its neighboring particles, which is accelerated by the widely-used hashing and sorting methods for SPH [IOS*14]. For spray simulation, we adopt the parallel CG method for Poisson equation, while using the same scheme as [Har04] for the semi-Lagrangian advection of density and velocity fields.

Transitions. Since we generate droplets and transit them to spray dynamically, the particle data should be frequently added and deleted, so we employ a memory compacting method for the CUDA-based memory structure for storing particle properties (such as mass, position, and velocity) when it is not consecutively placed. After introducing this compacting method, we detail the implementation of transitions.

Compact CUDA Memory. As shown in Fig. 3, array A indicates whether a particle should be preserved or not after compaction, wherein 1 means it will be preserved. In this case, we use the thrust library in CUDA SDK to scan A and obtain array B, of which $B[i]$ indicates that particle p_i should be moved to place $B[i]$ in memory compaction. Array C and D show the corresponding relationship of the particle data in the compaction.

Generate Droplets. We pre-allocate a large enough memory space on GPU so that each particle can occupy n_{max} consecutive memory space, where n_{max} denotes the maximal droplet particles that one liquid particle can split into. When generating droplets, we first invoke a CUDA kernel for each liquid particle i, and then check whether it satisfies the generation criterion (Section 4.1.1) or not. If so, we randomly generate the masses and positions of droplets, and write them into the memory space indexed by $i * n_{max} + j$, where j denotes the j-th droplet particle corresponding to the liquid particle i. Meanwhile, we maintain an array to indicate the practical space usage of the particle array (Fig. 3), and subsequently compact the memory space.

Merge Liquid Particles. We invoke a CUDA kernel for each liquid particle p_i to search and record the nearest neighboring particle p_j that satisfies (1) $m_i + m_j < m_{max}$, (2) $d_{ij} < r_i + r_j$, where r denotes the particle radius and m_{max} is the maximum mass allowed in our implementation. Suppose p_j is the nearest particle recorded for p_i, it should be noted that p_i is not always the nearest one for p_j. And we merge particle p_i and p_j if and only if they record each other as the nearest particle. In merging step, we also maintain the memory space usage array (array A in Fig. 3) and compact memory space accordingly.

Transit from Droplets to Spray. To implement the secondary atomization, we firstly invoke a CUDA kernel for each droplet particle, in which we check if it satisfies the criterion in Section 4.3. After the transition mass is determined, we search all the neighboring grid cells, and add the weighting mass onto them with the *atomicAdd* operator defined in CUDA API.

7. Experimental Results and Discussion

Our hybrid particle-grid model has been implemented with C++ and CUDA, and all the examples are run on a PC with Geforce GTX 780 GPU and Intel Core i7 CPU. All of our results are rendered by POV-Ray (http://www.povray.org). Table 1 shows the default parameters used in our experiments and Table 2 lists the experimental performance, indicating the high efficiency of our CUDA-based hybrid model implementation. The computational time for liquid and droplets depends on both particle number and grid resolution, while spray simulation time only depends on the grid resolution. It may be noted that the waterfall scene with lower grid resolution instead takes more time because of the time-consuming collision detection and response between liquid/droplet particles and the terrain.

Fig. 4 shows a fountain example, where four violent streams are spouted into the air. As they rise and the relative velocities of liquid particles $\mathbf{v}^{rel}$ increase, more disperse droplets are generated, and subsequently spray is formed from the secondary atomization of droplets. The remaining droplets fall down into the water tank, and arouse disturbance on the water surface. When an extra wind is enforced, the droplets and spray are blown away due to the drag force.

Fig. 5 demonstrates a scene where the water stream inter-

Table 1: *Parameter values used in our experiments.*

γ_{th}	$\mathbf{v}_{th}^{rel}$	c	$(\beta, v_{min}, v_{max})$ (Eq. 9)	(ρ_{min}, ρ_{max}) (Eq. 15)	(μ, d) (Eq. 18)
16.0	0.5	0.1	(0.01, 1.5, 3.5)	(200, 1000)	(0.01, 0.005)

Table 2: *Time performance (in milliseconds) of our experiments.*

Scene	Grid Resolution	#Particles (Liquid, Droplet)	FLIP	Droplet	Spray	Transition
Fountain (Fig. 4)	$96 \times 64 \times 96$	(121k, 95k)	23.2	26.5	31.5	19.2
Interaction (Fig. 5)	$96 \times 64 \times 64$	(15k, 10k)	12.0	16.1	28.9	11.2
Waterfall (Fig. 6)	$64 \times 64 \times 64$	(57k, 81k)	30.5	39.9	22.1	16.6
Nozzle (Fig. 7)	$128 \times 64 \times 64$	(0, 4.9k)	0	22.1	36.9	10.1

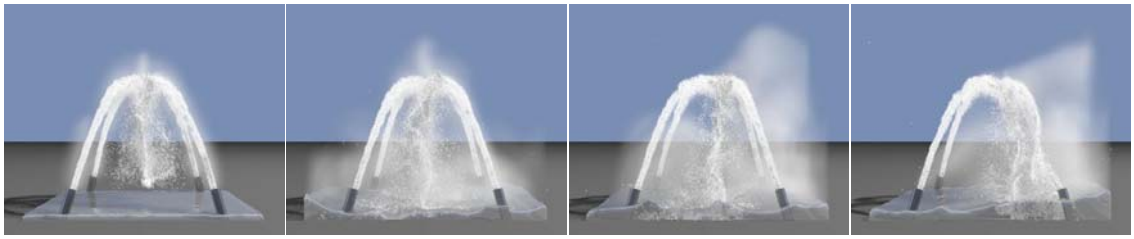

Figure 4: *Fountain. Water is spouted into the air, and a wind is enforced in the third and fourth images.*

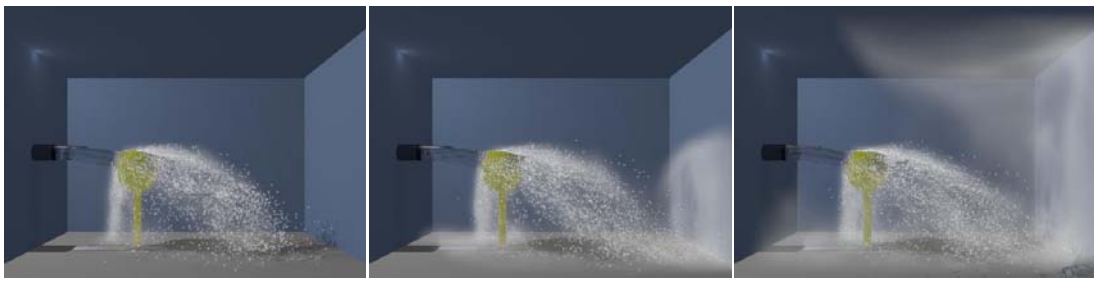

Figure 5: *Interaction effects. Liquid is pouring onto a static sphere supported by a stick to show the droplets/spray mixture due to collision.*

Figure 6: *Waterfall. Liquid stream is pouring down into a tank of water, droplets and spray are generated when the falling water becomes rapid and violent, especially when it hits the rocks.*

acts with a static sphere supported by a stick. The high speed stream becomes violent after collision, disperse droplets and spray are produced at the same time. As more liquid is converted to spray, the mist fills the room gradually. It may be noted that, after collision with sphere, most of the water is transited to droplets, this phenomenon can not be reasonably generated by solely relying on existing post-processing methods [GLR*06, FGP07, IAAT12].

Fig. 6 shows a waterfall scene. At the top of waterfall, the speed of water is low, as the water stream falls down rapidly and the air-liquid relative velocity $\Delta\mathbf{v}$ increases, various droplets and spray are generated. While at the bottom, the falling water rushes into the water tank and crashes on the terrain, the high relative velocity results in splashing droplets and fog-like spray. In comparison with other experiments, the computational cost is heavier due to the collision detection and response with the employed terrain.

To demonstrate the secondary atomization phenomenon, a nozzle example is shown in Fig. 7, where we directly emit droplet particles and only enable the transition from droplets to spray. From top to bottom, we gradually increase the relative air-droplet velocity threshold (v_{min}, v_{max}) in Fig. 7(a, b, c) to make the droplets-spray transition more obvious. Meanwhile, in Fig. 7(d) we ignore the mass transport in the same way as the existing methods [TFK*03, KCC*06, MMS09] do, the masses of droplets remain unchanged even when spray is generated, which can not simulate the phenomenon of completely transfering droplet to spray and subsequently dissipating into the air.

In Fig. 8 we render the droplets and liquid as particles, and code the color according to radius, demonstrating the effect of particle distribution. At the top of the fountain in the right image, the large droplets become smaller and some of them disappear due to secondary atomization, resulting in dense spray as shown in Fig. 4. Meanwhile, the smaller droplets are slowed down by drag force more quickly, resulting in falling down earlier than bigger droplets.

Limitation. Even though the strategy of coupling different models offers us more flexibility to tune the simulation results, the most severe limitation of our method is the complexity. We couple three types of simulation methods to synchronously accommodate liquid, droplets, and spray simulation. It undoubtedly increases the computational cost. The required neighboring search of particles also increases computational cost. Meanwhile, our method handles the droplet collision via an Eulerian model and ignores the shapes of droplets and their precise inter-collision, which makes it appropriate for large-scale violent liquid simulation, but not suitable for small-scale simulation whose goal is to concentrate more on the deformation and topological change of droplet. Besides, since this paper focuses on the GPU-based interactive numerical solver of our proposed hybrid model, we choose to directly employ the famous POV-Ray to serve as an off-line renderer, nevertheless, such renderer can be replaced with other real-time and more realistic rendering routines/plug-ins without any difficulty.

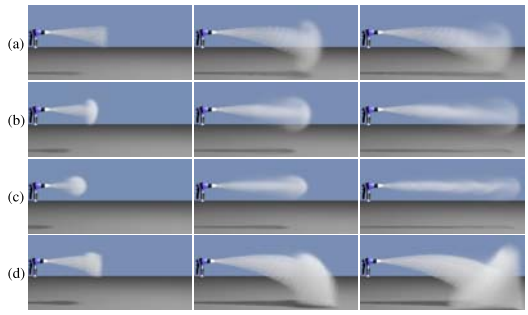

Figure 7: *Nozzle. Initial velocity is set to be 9.0m/s. From top to bottom, (v_{min}, v_{max}, β) in Eq. 9 are respectively set to be (4.5, 6.5, 0.05), (7.5, 8.5, 0.1), (7.5, 10.5, 0.1), and (7.5, 8.5, 0.1). (a, b, c) consider the mass transport from droplet to spray, which is ignored in (d).*

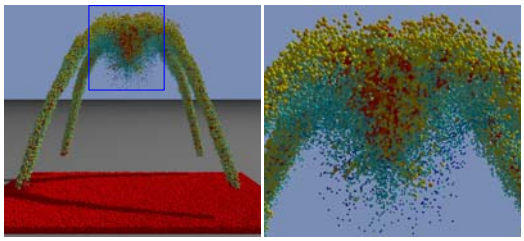

Figure 8: *The particles of liquid and droplet in fountain scene are color-coded according to radius, where blue means small and red means large.*

8. Conclusion and Future Work

We have detailed a novel hybrid particle-grid method to simulate violent liquid stream with massive disperse droplets and fog-like spray. By simultaneously considering transitions among liquid, droplets and spray, our model can produce the rich visual effects of splashing liquid and mist. Benefitting from the the hybrid of Lagrangian and Eulerian methods, the complex dynamics of droplets can be simulated effectively. Meanwhile, adaptive multi-scale droplets and their momentum exchange with the surrounding air improve the versatility and flexibility of the simulation results. Moreover, for a scenario containing 200,000 particles on a $96 \times 64 \times 96$ grid, our CUDA-based implementation can achieve interactive performance while still exhibiting very promising visual effects.

At present, we choose to ignore the spray-droplet transition for efficiency purpose, we shall continue to study the spray condensation mechanism and involve other environmental factors such as temperature, floating dust, etc, so that

our transition model could accommodate a more physics-meaningful closed loop. Besides, developing a built-in, realistic rendering framework for our hybrid model also deserves further investigation, which could broaden the practical spectra of our method and its utility.

Acknowledgements: This research is supported by National Natural Science Foundation of China (No.61190120, No.61190125, No.61190124, and No.61300067), NSF (IIS-0949467, IIS-1047715, and IIS-1049448), and China Scholarship Council (CSC). We also thank the anonymous reviewers for their constructive critiques.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July 2007), 481–487. 2

[ATT12] ANDO R., THUREY N., TSURUNO R.: Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (Aug 2012), 1202–1214. 2, 3, 4, 5

[BB12] BOYD L., BRIDSON R.: MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2 (Apr. 2012), 16:1–16:12. 2

[CCLW11] CORRIGAN A., CAMELLI F. F., LÖHNER R., WALLIN J.: Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids 66*, 2 (2011), 221–229. 3

[CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: IISPH-FLIP for incompressible fluids. In *Computer Graphics Forum* (2014), vol. 33, pp. 255–262. 2

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Proc. of SCA'10* (2010), pp. 197–206. 2

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (July 2002), 736–744. 2

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), SIGGRAPH '01, ACM, pp. 23–30. 2

[FGP07] FROEMLING E., GOKTEKIN T., PEACHEY D.: Simulating whitewater rapids in ratatouille. In *ACM SIGGRAPH 2007 Sketches* (2007), SIGGRAPH '07, ACM. 3, 9

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 185:1–185:6. 2

[GLR*06] GEIGER W., LEO M., RASMUSSEN N., LOSASSO F., FEDKIW R.: So real it'll make you wet. In *ACM SIGGRAPH 2006 Sketches* (2006), SIGGRAPH '06, ACM. 3, 9

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proc. of SCA'10* (2010), pp. 55–64. 3

[Har04] HARRIS M.: Fast fluid dynamics simulation on the gpu. *GPU gems 1* (2004), 637–665. 3, 7

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. *Proc. of Computer Graphics International* (2007), 63–70. 3

[IAAT12] IHMSEN M., AKINCI N., AKINCI G., TESCHNER M.: Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput. 28*, 6-8 (June 2012), 669–677. 1, 2, 9

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core CPUs. *Comput. Graph. Forum 30*, 1 (2011), 99–112. 3

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014-State of the Art Reports* (2014), The Eurographics Association, pp. 21–42. 2, 7

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *Proc. of SCA'06* (2006), pp. 335–344. 1, 2, 3, 9

[KE12] KROG Ø. E., ELSTER A. C.: Fast GPU-based fluid simulations using SPH. In *Applied Parallel and Scientific Computing*, vol. 7134. 2012, pp. 98–109. 3

[LTKF08] LOSASSO F., TALTON J., KWATRA N., FEDKIW R.: Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (July 2008), 797–804. 2, 3

[LWGP08] LIU S., WANG Z., GONG Z., PENG Q.: Simulation of atmospheric binary mixtures based on two-fluid model. *Graphical Models 70*, 6 (2008), 117 – 124. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of SCA'03* (2003), pp. 154–159. 2, 5

[MDCN03] MIZUNO R., DOBASHI Y., CHEN B.-Y., NISHITA T.: Physics motivated modeling of volcanic clouds as a two fluids model. In *Pacific Conference on Computer Graphics and Applications* (2003), IEEE, pp. 440–444. 2

[MDN02] MIYAZAKI R., DOBASHI Y., NISHITA T.: Simulation of cumuliform clouds based on computational fluid dynamics. *Proc. Eurographics 2002 Short Presentation* (2002), 405–410. 2

[MMS09] MIHALEF V., METAXAS D., SUSSMAN M.: Simulation of two-phase flow with sub-scale droplet and bubble effects. *Comput. Graph. Forum 28*, 2 (2009), 229–238. 1, 2, 4, 5, 9

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 2

[NØ13] NIELSEN M. B., ØSTERBY O.: A two-continua approach to eulerian simulation of water spray. *ACM Trans. Graph. 32*, 4 (July 2013), 67:1–67:10. 1, 2, 5

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Trans. Graph. 30*, 4 (Aug. 2011), 81:1–81:8. 2

[Sir10] SIRIGNANO W. A.: *Fluid dynamics and transport of droplets and sprays*. Cambridge University Press, 2010. 4

[Sta99] STAM J.: Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (1999), SIGGRAPH '99, pp. 121–128. 6

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Comput. Graph. Forum 22*, 3 (2003), 391–400. 1, 2, 3, 9

[TRS06] THÜREY N., RÜDE U., STAMMINGER M.: Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proc. of SCA'06* (2006), pp. 157–164. 2

[YWH*09] YAN H., WANG Z., HE J., CHEN X., WANG C., PENG Q.: Real-time fluid simulation with adaptive SPH. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 417–426. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972. 2, 3