A Novel Integrated Analysis-and-Simulation Approach for Detail Enhancement in FLIP Fluid Interaction

Lipeng Yang, Shuai Li,* Qing Xia, Hong Qin, and Aimin Hao

Abstract

This paper advocates a novel integrated method to tightly couple simulation with analysis for the effective modeling and enhancement of scale-aware fluid details. It brings forth a suite of innovations in a unified framework, including depth-image-based space analysis for multi-scale detail detection, time-space analysis based on the logistic regression model that integrates both geometry and physics criteria, and depth-image-based sampling for quality-efficiency tradeoff. Our method contains an intertwined two-level processing architecture at its core. At the analysis level, we propose a rigorous time-space analysis model to pinpoint complex interacting regions, which can take into account multiple detail-relevant factors based on the depth-image sequence captured from FLIP-driven simulation sequence. At the simulation level, details are enhanced by animating extra diffuse materials, and augmenting the air-fluid mixing phenomenon. Directly benefitting from the flexibility of image-space-dominant processing, our unified framework can be entirely implemented on GPU, hence interactive performance could be guaranteed. Comprehensive experiments and evaluations on various diffuse phenomena (e.g., spray, foam, and bubble) have demonstrated its superiority in high-fidelity detail enhancement during fluid simulation and its interaction with surrounding environment for VR applications.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling—;

Keywords: Fluid Detail Enhancement, Time-space Analysis Model, Image Space Method, FLIP, GPU

1 Introduction and Motivation

During the past two decades, physically-based fluid simulation has become a rapidly-evolving research topic in many virtual reality (VR) applications, wherein various methods have been developed in response to different motivations, including single phase fluid simulation, multiple fluid interaction, two-way fluid-solid coupling, etc. With the growing demand for physical realism and interactive cross-scale detail enhancement, such as splash, spray, and other diffuse materials [Losasso et al. 2008; Ihmsen et al. 2012], the systematic and comprehensive studies of fluid detail simulation are beginning to regain popularity with great and revived momentum.

Although accuracy is the key concern in computational fluid dynamics, visual fidelity and efficiency are more important in graphics, animation and many other VR applications. For the visual fidelity improvement, the state-of-the-art methods oftentimes resort to various space-grid refinement or denser particle sampling schemes in physical simulation. Considering the unavoidable increase of computational expenses, adaptive sampling methods are more favorable, such as octree-grid [Losasso et al. 2004], adaptive SPH [Adams et al. 2007], etc. Meanwhile, parallel methods on multi-core CPUs or GPU are also developed in order to reduce the ever-increasing computational cost [Goswami et al. 2010; Krog and Elster 2012], while still improving visual quality. From the perspective of pure simulation, other key ideas to enhance details include handling different materials separately (such as air and liquid) and simulating the crucial interaction [Takahashi et al. 2003; Kim et al. 2006; Mihalef et al. 2009] in a physically-plausible fashion.

Despite the recent success of diffuse material simulation, certain difficulties still prevail and need to be addressed for high-fidelity region detection for fluid details in a more efficient way. First, most of the detail detection criteria depend on the correct identification of accurate fluid surface, which is too time-consuming to be precisely detected in any physics-driven real-time applications. Meanwhile, spatially-local analysis criteria, such as curvature and velocity variance, are usually combined manually with many parameters, which are hard to be fine tuned for different phenomena. Second, most of the criteria are hard to pinpoint scale-aware details, wherein the details are mostly determined by probability-biased parameters and their ad hoc combination. Third, the state-of-the-art analysis methods tend to ignore the informative time-domain knowledge, which is crucial for fluid detail detection and prediction.

Our rationale is that, pure physics-based simulation may fall short in fluid detail enhancement during interaction, unless significantly increased computational expenses could be afforded. Hybrid strategies, which integrate simulation and analysis, must be explored towards high-fidelity fluid interaction while not sacrificing computational efficiency drastically. This paper attempts to offer a viable solution to tackle the aforementioned technical challenges. We articulate a novel integrated analysis-centric approach by rigorously analyzing the acquired data. In particular, we are proposing to utilize a depth buffer sequence captured during simulation as an indicator of fluid surface. By analyzing the time-space features in image space, we expect to correctly identify the fluid-detail regions and then simulate the diffuse materials to augment the air-fluid interacting phenomena. Through the integration of simulation and analysis, we are hoping to achieve the competing goals of high-fidelity fluid detail enhancement and computational efficiency for VR applications. The salient contributions of this paper can be summarized as follows:

- We propose a time-space analysis model based on logistic regression to integrate the geometric, physical and temporal factors for fluid details detection while avoiding complex parameter tuning.
- We propose to bridge simulation and analysis via the utility of dynamically-captured depth buffer data, which is both low-cost and versatile for quality-efficiency tradeoff.
- We develop a seeding method for diffuse particle generation, which only relies on the depth buffer and analysis results, making it applicable to both particle and grid based simulation methods.
- We design specific GPU-based algorithms to implement our

*e-mail:lishuai@buaa.edu.cn, State Key Laboratory of Virtual Reality Technology and Systems, Beihang University.

VRST '15, November 13 – 15, 2015, Beijing, China.
© 2015 ACM. ISBN 978-1-4503-3990-2/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2821592.2821598

entire simulation framework in parallel.

2 Related Work

In recent years, there have been many methods developed to simulate fluids, including Smoothed Particle Hydrodynamics (SPH) method [Monaghan 1992; Müller et al. 2003; Ihmsen et al. 2014], Fluid Implicit Particle (FLIP) method [Zhu and Bridson 2005], and Level Set methods [Enright et al. 2002; Foster and Fedkiw 2001], etc. Our approach is based on FLIP method, which combines particle-based representation with grid-based solver. FLIP method was introduced in computer graphics by [Zhu and Bridson 2005], and then was extended to simulate splashing water [Gerszewski and Bargteil 2013], preserve fluid sheet [Ando et al. 2012], conduct fluid-solid coupling [Selino and Jones 2013], combine with particle method [Cornelis et al. 2014], model multi-scale droplet/spray [Yang et al. 2014], etc. Lately, [Ando et al. 2012] and [Cornelis et al. 2014] respectively proposes methods to improve the particle distribution of FLIP method.

Closely relevant to the central theme of this paper, we now briefly review previous works in diffuse material simulation and data-specific analysis.

Diffuse Material Simulation. Diffuse material simulation is crucial in fluid detail simulation and enhancement, especially for the large-scale fluid phenomenon having complex interaction with air.

Detecting diffuse materials is straightforward for grid-based methods, since the surface of liquid can be easily tracked. Existing works essentially measure geometric information to generate splash, foam, and bubble using curvatures [Takahashi et al. 2003], markers escaped from surface [Kim et al. 2006; Losasso et al. 2008; Hong et al. 2008], etc. To exploit the ignored velocity information, [Mihalef et al. 2009] presents a Weber number threshold based filtering method to avoid the detail loss of the marker level set method by introducing a physical factor into the criteria. Meanwhile, some other works regard air as a separate phase fluid and simulate the interaction between liquid and air directly, including volume-of-fluid [Hong and Kim 2003], regional level set [Zheng et al. 2006], MultiFLIP [Boyd and Bridson 2012], two-continua approach [Nielsen and Osterby 2013], etc. [Patkar et al. 2013] proposes a hybrid method for bubbles, using level set method to simulate large bubbles while employing particle method to represent small bubbles.

As for particle-based methods, analyzing the diffuse materials is not trivial, because the precise fluid surface is not easy to be detected and extracted. To simulate the water-air interaction, [Müller et al. 2005; Solenthaler and Pajarola 2008] dynamically generate air particles by analyzing the geometric properties of liquid surface, and propose a two-phase SPH model. Although being capable of obtaining improved visual results, their methods still have problems when the density ratio between air and liquid is large. To ameliorate, [Ihmsen et al. 2011] simulates the bubbles and liquids separately by differentiating the regions where air is likely trapped with a velocity-based heuristic criterion. [Bagar et al. 2010] detects foam particles from regular liquid particles based on the velocities of particles, which involves no foam simulation. To unify spray, foam, and air bubble, [Ihmsen et al. 2012] proposes a post-processing model to generate and advect the diffuse materials for SPH fluids, wherein the diffuse regions are detected by analyzing the curvatures and velocities of SPH particles. Although such scheme avoids expensive inter-particle computation, it heavily relies on the proper detection of surface particles and the curvature calculation over surface particles.

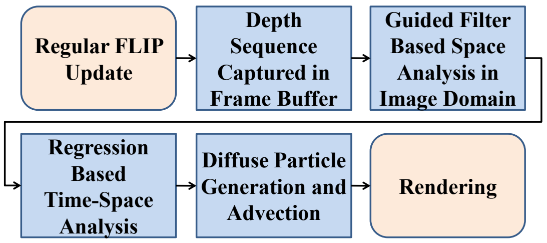
Regular FLIP Update
Depth Sequence Captured in Frame Buffer
Guided Filter Based Space Analysis in Image Domain
Regression Based Time-Space Analysis
Diffuse Particle Generation and Advection
Rendering

Figure 1: *The pipeline of each simulation cycle.*

Data-Specific Analysis Methods. Image-specific analysis methods are commonly used to enhance image details, which in our method will be employed to detect the complex interacting regions in depth-images. Most recently, some researchers produced plausible results in image smoothing and detail enhancement, such as data-specific wavelet [Hammond et al. 2011; Narang and Ortega 2013], L0 gradient minimization based smoothing [Xu et al. 2011], etc. However, the computational costs of these methods are expensive and not suitable for interactive VR applications. Guided image filtering is an optimization-based image filtering method proposed in [He et al. 2013], which can be exactly and efficiently computed. Compared with the widely used bilateral filter [Tomasi and Manduchi 1998], guided filter is more generic and has a better trade-off on accuracy and efficiency.

To integrate time-domain information into our method, we refer to time-space analysis, which is commonly utilized in video object tracking. The relevant techniques include Bayesian model [Cagniart et al. 2010; Khan and Gu 2011], logistic regression [Ciocca et al. 2013], Gaussian mixture model [Carmi et al. 2012], etc. Our detail region detection is similar with video tracking in the aspects that both time and space information are crucial, but our aim is to guarantee the continuous evolution of enhanced regions. It is appropriate to use the logistic regression to combine multiple time-space factors in order to obtain the probability of each pixel in depth buffer, indicating whether it should be enhanced.

3 Algorithmic Overview

Fig. 1 illustrates the entire algorithmic flow of our method. Each simulation cycle starts with regular FLIP, by way of analyzing the captured result we enhance fluid details via animating diffuse materials. We outline the algorithm as follows:

FLIP Update. Compute the density, pressure, force for each fluid particle, and then update its velocity and position.

Depth Capture. Capture and store the depth buffer dynamically from the top view point with an *orthogonal virtual camera*.

Space Analysis. Analyze the 2D depth buffer spatially with data-specific guided filter to obtain the geometric features.

Time-Space Analysis. Integrate temporal information and spatial analysis result into a logistic regression model to obtain the scale-aware criteria for detail enhancement.

Diffuse Particle Generation and Simulation. Generate diffuse particles based on the analysis results, then advect them according to the material-specific rules.

Rendering. Render the scene with POV-Ray software.

3.1 Brief FLIP Review

Since our model is based on the adaptive FLIP model [Ando et al. 2012], we now briefly review the basic idea of fluid simulation, FLIP model, and its possible improvement. Fluid dynamics are essentially based on Navier-Stokes equations (N-S equations) that conserve both mass and momentum:

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

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

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 instead. Unlike Particle-in-Cell (PIC) method, the velocity changes on grid, rather than computing the velocity on grid directly from the aforementioned equations, the grid velocity is interpolated from surrounding 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 [Zhu and Bridson 2005], we linearly blend the PIC and FLIP velocities via

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

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

Besides, the original FLIP method suffers from the uneven distribution of particles, which is often solved by re-sampling. To improve, [Ando et al. 2012] proposes a position correcting step, which is similar to the pressure step in SPH. Since re-sampling is avoided in [Ando et al. 2012], the particle number is kept unchanged in the simulation. Thus it can easily make the FLIP method retain SPH-like mass-conserving features. It should be noted that, the mass conservation is critical to our hybrid model together with the stability of the FLIP method for accommodating varying-scale particles.

4 Time-space Analysis

4.1 Depth Image Sequence Capture

As shown in Fig. 1, the depth buffer is captured at the end of each regular FLIP update cycle. Similar to [Van der Laan et al. 2009] (but with a downward *orthogonal camera* at the top of the scene), we render the particles as spheres, and then capture the depth buffer from frame buffer. The captured depth buffer is stored as texture first, and then is mapped into GPU memory as 2D array, which is the basis of conducting analysis. The depth buffer actually forms a 2D *projected grid* in simulation domain, which enables us to extract velocity field and seed diffuse particles. Meanwhile, we also keep a record of some sequential frame buffers to analyze the temporal information involved in the dynamics of the underlying fluid surface.

The resolution of captured depth buffer is crucial to the tradeoff of the analysis quality and computational cost. An appropriate resolution could accommodate sharp/fine features on the fluid surface while guaranteeing efficiency. We will discuss our resolution selection strategy and demonstrate the results of different resolutions in Section 7.

4.2 Guided Filter based Space Analysis

Geometric features of fluid surface is critical to determining where the details should be enhanced. By converting the fluid surface feature detection to a problem of depth image analysis, it can not only reduce the scale of computation but also take advantages of the abundance of techniques developed in image processing. We employ the content-specific guided filter [He et al. 2013] to perform detail detection. The key idea of guided filter is to develop a filter kernel that can be applied onto the input image to produce a new image. In our framework, we apply its smoothing function to obtain smoothed depth buffers, whose differences could pinpoint scale-aware detail region with the high-frequency characteristics.

We assume there is a local linear transformation between guidance image I and output q in a window w_k centered at the pixel k,

$$q_i = a_k I_i + b_k, \forall i \in w_k, \tag{4}$$

where a_k and b_k are linear coefficients. To determine a_k and b_k, we minimize the difference between q and the input image p,

$$E(a_k, b_k) = \sum_i ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2), \tag{5}$$

where ϵ is a regularization parameter, and we define the guidance image I and the input image p with the same depth buffer.

Then a_k, b_k can be directly solved by linear regression as

$$a_k = \frac{\frac{1}{|w|}\sum_{i \in w_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}, \tag{6}$$

$$b_k = \bar{p}_k - a_k \mu_k, \tag{7}$$

where μ_k and σ_k^2 are the mean and variance of I in w_k, $|w_k|$ is the number of pixels in window w_k, and $\bar{p}_k$ is the mean of p in w_k.

After that, q can be computed using Eq. 4. Since each pixel i belongs to all the windows centered at the surrounding pixels k, we average a_k, b_k to guarantee consistency.

$$q_i = \frac{1}{|w|} \sum_{k: i \in w_k} (a_k I_i + b_k) = \bar{a}_k I_i + \bar{b}_k, \tag{8}$$

where $\bar{a}_k$ and $\bar{b}_k$ are the mean of a_k and b_k in window w_i. We compute the differences of smoothed multi-scale depth buffers to obtain potential detail features, which reside in the high-frequency regions of the fluid surface (refer to Fig. 2(b)). The window size w_k allows us to identify features of different scales (refer to Fig. 7 and our supplementary video).

4.3 Regression-based Time-space Analysis

Besides the aforementioned geometric features, we further exploit velocity field as a physical criterion to determine the detail region. Specifically, we only need the velocities at the projected grid nodes, which can be extracted by finding the fluid particles around each node and computing the weighted sum of the fluid velocities. Fig. 2(c) shows the examples of the extracted velocity field.

Once we obtain the geometric and velocity information of depth image, we need to combine them together to form a criterion, which indicates the regions where diffuse materials are prone to generate. To complete this task, we explore the logistic regression model, which is a classic machine learning model to estimate the probability of binary response based on one or more predictor variables. In our method, logistic regression model takes the

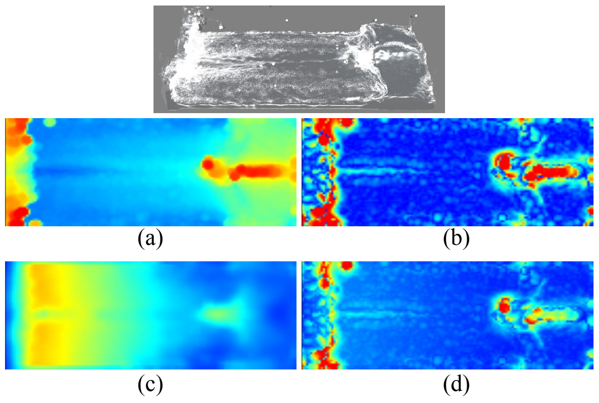

Figure 2: *Time-space analysis results. (a) The captured depth buffer, (b) The space analysis result, (c) Velocity field, (d) Time-space analysis result.*

geometric and physical information as inputs, and generates a probability for each pixel in depth image, indicating its probability of generating diffuse material. We define a logistic function to compute the probability that a depth buffer pixel should fall in a detail region: $h_\theta(x) = 1/(1 + exp(-\sum_{j=1}^{n} \theta_j x_j))$, where x_j represents all the factors (including velocity field v, which could be obtained from the spatial analysis result) taken into consideration, and θ_j is the corresponding coefficient to be learned.

The same as the classic logistic regression model, we use maximum likelihood estimation to learn the coefficients θ_j, so the cost function in our model is as follows,

$$E_s = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log h_\theta(x^{(i)}) + (1- \\ y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + r||\theta||^2. \tag{9}$$

Here m denotes the number of samples, which is the number of pixels in the image, $y^{(i)}$ denotes the probability of i-pixel position that should be enhanced, and r is a user-defined coefficient to prevent θ from being too large. By minimizing the E_s in Eq. 9 using gradient descent method, we can obtain the coefficients θ_j.

However, in the temporal aspect, the features of fluid are changing continuously, we expect the adjacent detected features would not differ too much. So we introduce a regularization penalty item to measure the inconsistency between adjacent frames, $E_t = \sum_{i=1}^{m}(y^{(i)} - y^t)^2$, where y_t is the analysis result of the previous frame.

By integrating the above aspects, the proposed cost function is formulated as follows:

$$E = E_s + \mu E_t, \tag{10}$$

where μ is a non-negative trade-off parameter. Considering y and θ are the independent variables to be learned from Eq. 10, we employ the gradient descent approach to minimize the cost function, which can be formulated as follows:

$$\theta_j^{n+1} = \theta_j^n - \alpha_1\frac{\partial}{\partial\theta_j}E = \theta^n - \alpha_1[\frac{1}{m} \\ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + 2r\theta_j], \tag{11}$$

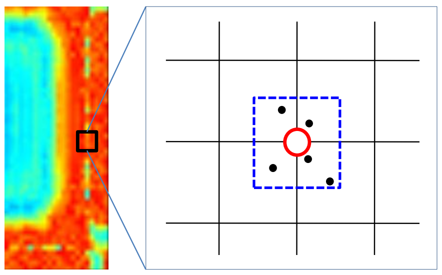

Figure 3: *Diffuse particle seeding.*

$$y^{n+1} = y^n - \alpha_2\frac{\partial}{\partial y}E = y^n - \\ \alpha_2(-\sum_{j=1}^{n}\theta_j x_j + 2\mu(y^n - y^t)). \tag{12}$$

In each simulation step, y and θ are initialized with the result from the previous step, while at the first step, y is set as 0, and θ is 1. In all experiments, r is 0.01, μ is 5.0, α_1 is 0.1, and α_2 is 0.01. By iterating θ and y until they converge, we obtain y to finally indicate whether a pixel of depth buffer (i.e., a node on the projected grid) is in the detail regions or not. The algorithmic details are shown in Algorithm 1 and Section 6. Fig. 2(d) shows our time-space analysis results as an example.

5 Diffuse Material Simulation

After determining the detail regions by time-space analysis, we seed diffuse particles correspondingly to augment the complex air-fluid interaction, and then advect them with the flow of fluid. This simulation strategy is similar to [Ihmsen et al. 2012], but we have modified the method to accommodate FLIP framework and GPU acceleration.

5.1 Seeding of Diffuse Particles

To be consistent with the time-space analysis result, we seed diffuse particles directly around the nodes of grid projected by depth buffer. For each projected grid node, we calculate the number of diffuse particles to be seeded and their 3D positions in the simulation space, and then seed the diffuse particles randomly around the projected grid node. Fig. 3 illustrates this process, the red point represents a pixel in depth buffer (i.e., a node on the projected grid), the square with dotted line shows the spatial range of seeding, and the dark points denote the generated diffuse particles. We utilize the depth image to generate diffuse particles, rather than relying on the particle distribution [Ihmsen et al. 2012], which reduces the computational time.

The newly generated diffuse particles are given an initial velocity $\mathbf{v}_{dp}$ to flow with the surrounding fluid particles. Meanwhile, we add a small random disturbance to avoid uniform movement of the diffuse particles via $\mathbf{v}_{dp} = \mathbf{v}_f + \mathbf{v}_{random}$, where $\mathbf{v}_f$ denotes velocity of nearby fluid, $\mathbf{v}_{random}$ is the disturbing velocity.

Meanwhile, to efficiently model the dissolution of the diffuse materials, we initially set a life time $t_{life} = ||\mathbf{v}_f||$ for each diffuse particle, which is related to the velocity of surrounding fluid particles $\mathbf{v}_f$. In each simulation step, we decrease t_{life} when the diffuse particle is classified as a foam particle, and if $t_{life} \leq 0$,

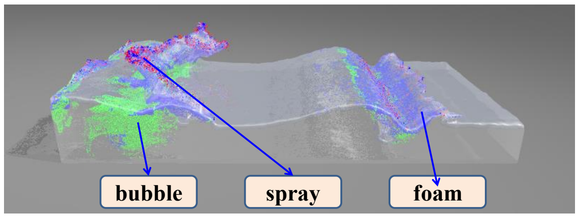

Figure 4: *The classification of diffuse particles (red for spray particles, green for bubble particles, and blue for foam particles).*

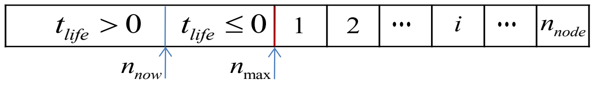

Figure 5: *The data structure to store the dynamic array of diffuse particles.*

we remove the particle. The diffuse particle classification will be described in Section 5.2. Although this lifetime model is simple to be implemented and can reflect the crack of bubbles and foams, but it ignores the complex physical factors such as temperature, size and material, which still may bring some artifacts in the simulation.

5.2 Advection of Diffuse Particles

To advect diffuse particles, we classify them into spray, foam, and bubble, which correspond to different advection formulations. This can be realized by computing the liquid particle number in the neighboring volume. In our experiments, when a diffuse particle is apart from the fluid surface and flying in the air, it is empirically classified as spray particle. When the particle is floating around the surface, it is classified as foam particle. While the particle is under the fluid surface, it is regarded as bubble particle. Fig. 4 illustrates a result of this classification method.

Air bubbles mean that air is trapped inside the liquid, which are mainly affected by buoyancy force and drag force from liquid particles. The drag force is determined by the relative velocities between the bubble particles and the liquid particles, the buoyancy force is enforced in the opposite direction of gravity acceleration $\mathbf{g}$.

$$\mathbf{f}_{dp} = k_d(\mathbf{v}_{dp} - \mathbf{v}_f) - k_b\mathbf{g}, \tag{13}$$

where k_{dp}, k_b correspond to the coefficients of drag force and buoyancy force, the subscript dp is always associated with a variable of diffuse particle.

Foam particles represent the foam at the surface of fluid, they are mainly affected by the drag force $k_d(\mathbf{v}_{dp} - \mathbf{v}_f)$ due to the flowing of liquid. Spray particles indicate the liquid particles that depart from the liquid volume, and they are only affected by the gravity force $m_{dp}\mathbf{g}$. Meanwhile, all the diffuse particles will be affected by a coupling force if they collide with solids in the scene. At last, the velocity and position can be updated based on Euler Integration.

6 CUDA-based Implementation

Since FLIP based simulations have been implemented on GPU efficiently [Yang et al. 2014], in the interest of space we only detail GPU implementation for the depth buffer based analysis and diffuse particle seeding/advection, which collectively guarantee the interactive performance of our framework. Algorithm 1 documents the pseudocode, and we detail the implementation challenges and solutions as follows.

Algorithm 1: Analysis-based Detail Enhancement

```
1  capture the depth buffer $d$ directly from scene.
2  smooth $d$ with guided filter method.
3  compute the difference $dif$.
4  compute the velocity field $v$.
5  initialize y, $\theta$;
6  while not converged do
7      while not converged do
8          update $\theta$ with Eq. 11
9      while not converged do
10         update $y$ with Eq. 12
11 seed diffuse particles at the projected grid node according to $d$ and $y$.
12 partition the array by thrust library.
13 advect all the diffuse particles with positive $t_{life}$.
```

Guided Filter based Space Analysis. Guided filter method is suitable for GPU parallelization, mainly because each pixel is only related to its surrounding pixels. We invoke a CUDA kernel for each pixel, compute ω_i, μ_i according to the surrounding pixels, and then compute a_k, b_k and store them in global memory cache. We invoke another kernel to compute the output image q and compute the difference simultaneously.

Regression-based Time-space Analysis. The key steps of time-space analysis are shown on line 5-10 of Algorithm 1. We need two kernel functions for line 8 and line 10, both of them will invoke a kernel thread for each pixel's parallel computation. In the first kernel function, we compute $\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$ with the parallel sum reduction, and then update θ and verify whether the convergence condition is satisfied. In the second kernel function, we update y for each pixel in a parallel way. When the computation is completed, the analysis result y is stored in the GPU array to be used in the following steps.

Diffuse Particle Seeding. Since diffuse particles are dynamically generated from projected grid according to the analysis results, we must design an appropriate data structure management on GPU to improve the algorithmic efficiency. We first estimate a maximum number of diffuse particles n_{seed} that each projected grid node may produce, and also estimate a maximum number of the diffuse particles n_{max} with positive t_{life} to be handled simultaneously in the animation. Then we allocate memory for $n_{seed} * n_{node} + n_{max}$ diffuse particles as shown in Fig. 5, where n_{node} denotes the number of projected grid nodes.

In each simulation cycle, we invoke a CUDA thread for each node of the projected grid, in which we produce diffuse particles around the node (refer to Fig. 3), and further set the velocities and life time respectively. The generated diffuse particles are dynamically inserted into the data structure as shown in Fig. 5, forming a sparse array. We then employ the thrust library to partition the array, moving the particles with positive duration time to the head of the array. At last, we invoke a thread for each particle to count the valid diffuse particles n_{now}, which is required in advection and rendering steps.

Diffuse Particle Advection. For diffuse particle advection, we invoke one thread for each particle to update the force, velocity, and position. Specifically, we utilize the fast neighborhood searching algorithm to find the surrounding fluid particles of diffuse particles, and then compute the average velocity of the neighboring fluid

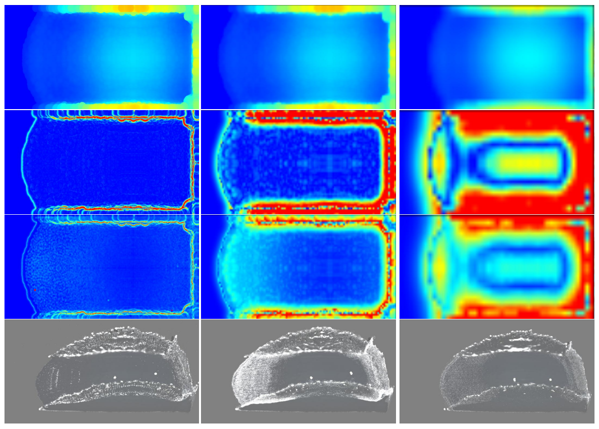

Figure 6: *Results corresponding to different resolutions. The resolution in simulation space is 140×60, and the depth buffer resolutions are: (a) 700×300, (b) 140×60, (c) 70×30.*

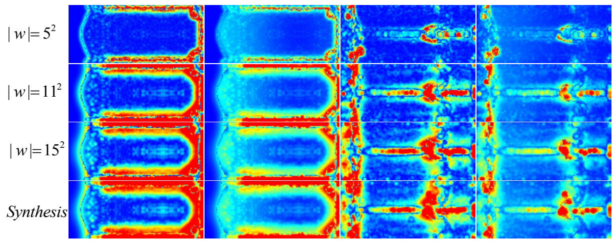

Figure 7: *Our scale-aware analysis results.*

particles (by using Eq.13). Meanwhile, we update the life time of diffuse particle in this thread. If the life time of a diffuse particle is negative, it will be overwritten after the partitioning operation, without explicitly deleting it.

7 Experiments and Evaluations

We have implemented our method on a PC with a Geforce GTX 780 GPU, Intel Core I7 CPU based on C++, CUDA, and GLSL APIs. For the issue of determining the resolution of depth buffer, we find the best analysis result is achieved when each pixel of depth buffer represents almost one particle, which means the width of depth buffer can be computed by $w_{db} = w_{scene}/(d_p)$, wherein w_{scene} denotes the width of scene and d_p is the diameter of fluid particle. From top to bottom, Fig. 6 shows the results of the captured depth buffer, guided filter based space analysis, time-space analysis, and simulation. Column (a) shows that higher-resolution results fail to represent the required local geometric information, while only reflecting the sphere shapes of particles in rendering. Meanwhile, the computational cost is more expensive compared with (b), which is the best in our experiments. Column (c) shows lower-resolution results, where the surrounding pixels involve analysis across a long distance in the simulation space. To handle details of different scales simultaneously, we detect guided filter features with different window sizes ($|w_k| = 5^2$, 11^2, or 15^2), then synthesize the corresponding results together as the final detected features. As shown in Fig. 7, our result reveals scale-aware features effectively.

Table 1 lists the statistics for the average testing time (in milliseconds) of each simulation cycle. We document the number of FLIP particles (#FP) and the maximum number of diffuse particles (#DP MAX) for each case. To be clear, (DP) is the time spent on diffuse particle generation and advection, (DB resolution) refers to *depth buffer resolution*, and (*Other*) is the time for deformable solid simulation and fluid-solid coupling. In comparison with the computational time of FLIP simulation, the overhead of our time-space analysis and diffuse material simulation is small, while improving the visual effects significantly. Table 2 documents the parameter values used in our experiments.

Table 2: *Parameter values used in our experiments.*

Parameters	Values
ϵ (Eq.5)	0.01
μ (Eq.10)	5.0
r (Eq.9)	0.01
α_1 (Eq.11)	0.1
α_2 (Eq.12)	0.01
k_d (Eq.13)	1.0
k_b (Eq.13)	5.0

Fig. 8 shows the simulation results of a dam-breaking scenario. In this scene, about 10k diffuse particles are generated nearby the wave crack, indicating the spray, foam, and small bubbles. As the fluid moves fiercely, the diffuse particles move correctly along with the wave. We can observe the scale of the generated diffuse particles closely relates to the height (determining the value of depth buffer at each grid) and velocity of the fluid.

Our method is also effective for a fluid-solid coupling scene, as shown in Fig. 9. A sphere is moving periodically on the fluid surface, pushing water aside and forming a small wave. Diffuse particles are generated according to the analysis result, these foams represent the trapped air in water, showing the trail of the sphere's movement, such details can not be captured without diffuse materials (as shown in the second row). The third row shows the involved different types of diffuse particles, which mainly represent foams (colored in blue).

Fig. 10 demonstrates a pouring scene, where four violent streams are spouted into the water tank. As the streams move, numerous spray particles are generated to enhance the visual details.

Fig. 11 shows a waterfall scene. As the water stream falls down rapidly, the analysis result indicates complex interaction happens, wherein the diffuse materials are generated due to the complex interaction with the underlying terrain. Comparing with the result without diffuse particle simulation (refer to the second row), our method enhances the simulation plausibility by adding spray, bubble, and foam. This scene also demonstrates the effectiveness of our analysis for large-scale simulation. In comparison with [Yang et al. 2014], our method ignores the smoke-like sprays, nonetheless, we are able to simulate spray, bubble, and foam in a unified framework, and our method is more efficient.

Fig. 12 shows the comparison between our method and that in [Ihmsen et al. 2012]. Comparing with [Ihmsen et al. 2012], our method defines completely different criteria to facilitate diffuse material generation, and can achieve competitive visual effects. It may be noted that, [Ihmsen et al. 2012] requires extracting surface particles and computing the local curvature from the neighboring particles, which is more suitable for densely sampled fluids. In sharp contrast, our method does not rely on the particles' information when computing the fluid surface curvature and other features. Directly benefitting from our time-space analysis, the results evolve continuously during the simulation (please refer to our supplementary video). Moreover, our method has a significant

Table 1: *Time performance (in milliseconds) of our experiments.*

Scene	#FP	Grid resolution (FLIP)	#DP MAX	DB resolution	FLIP	Analysis	DP	Other
Dam-break (Fig. 8)	248.4k	96×48×64	288×144	108.2k	47.7	20.2	5.3	-
Interaction (Fig. 9)	355.3k	96×48×64	288×144	49.6k	54.4	23.2	5.5	0.6
Pouring (Fig. 10)	242.1k	96×64×64	288×192	38.7k	41.5	18.4	5.3	-
Waterfall (Fig. 11)	212.9k	64×64×64	192×192	40.8k	40.8	16.6	5.2	36.9

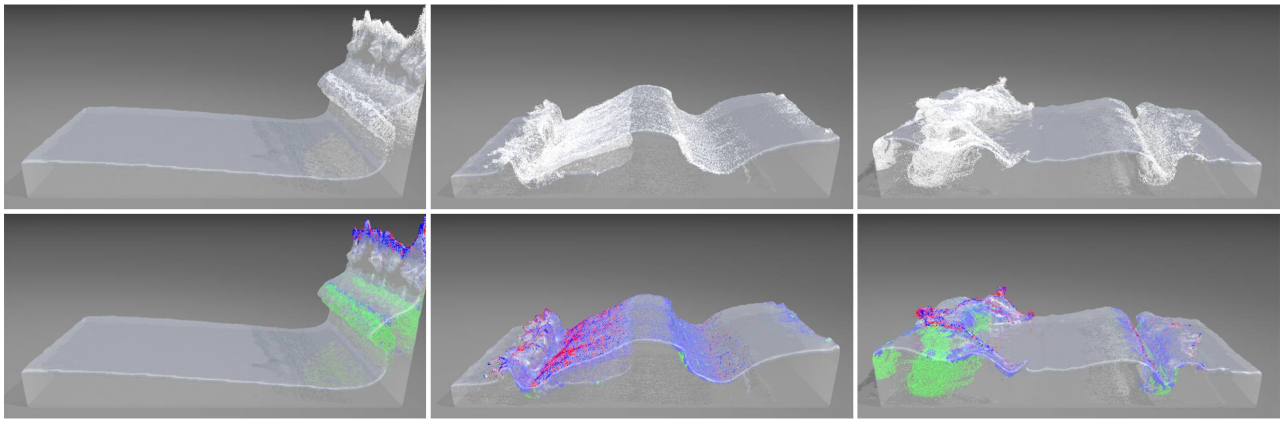
Figure 8: *Dam-breaking simulation. The first row shows the diffuse material simulation result, while the second row shows the classification of diffuse particles.*

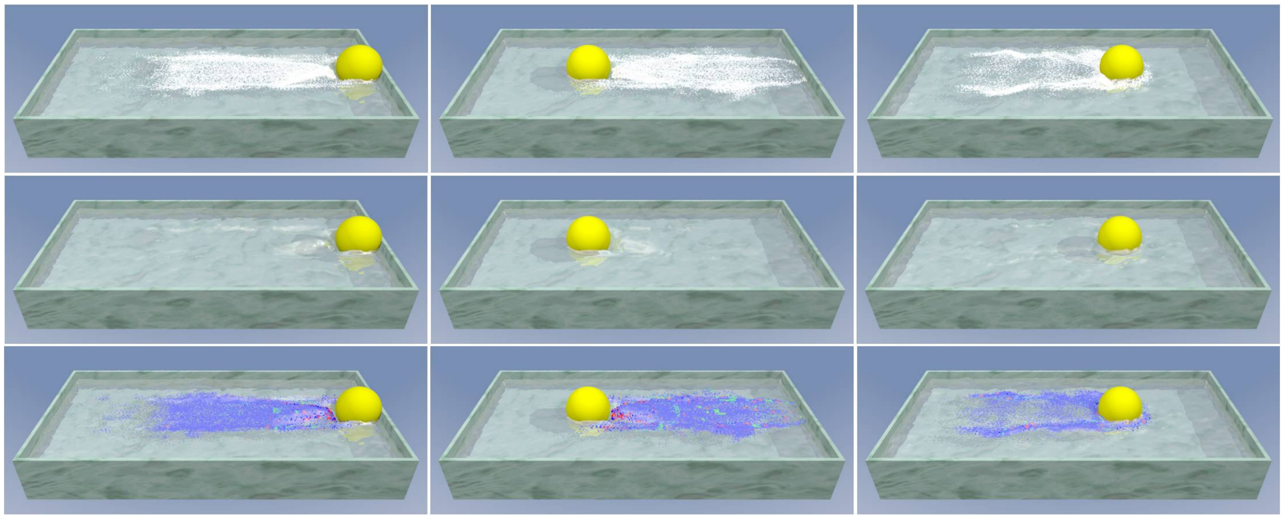
Figure 9: *The interaction effects between solid and water. The first row shows simulation result with diffuse materials, the second row shows the original FLIP simulation result, and the third row shows the classification of diffuse materials.*

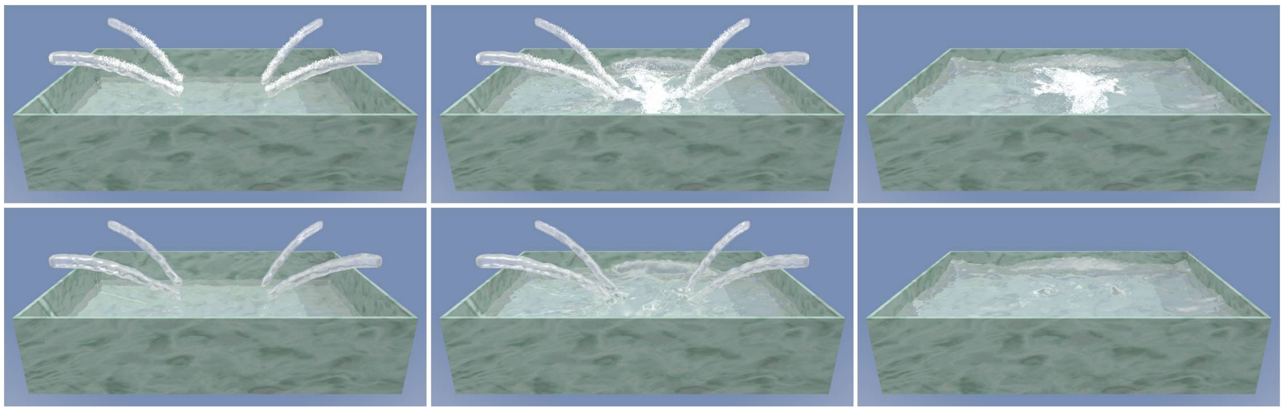

Figure 10: *Pouring water into a water tank.*

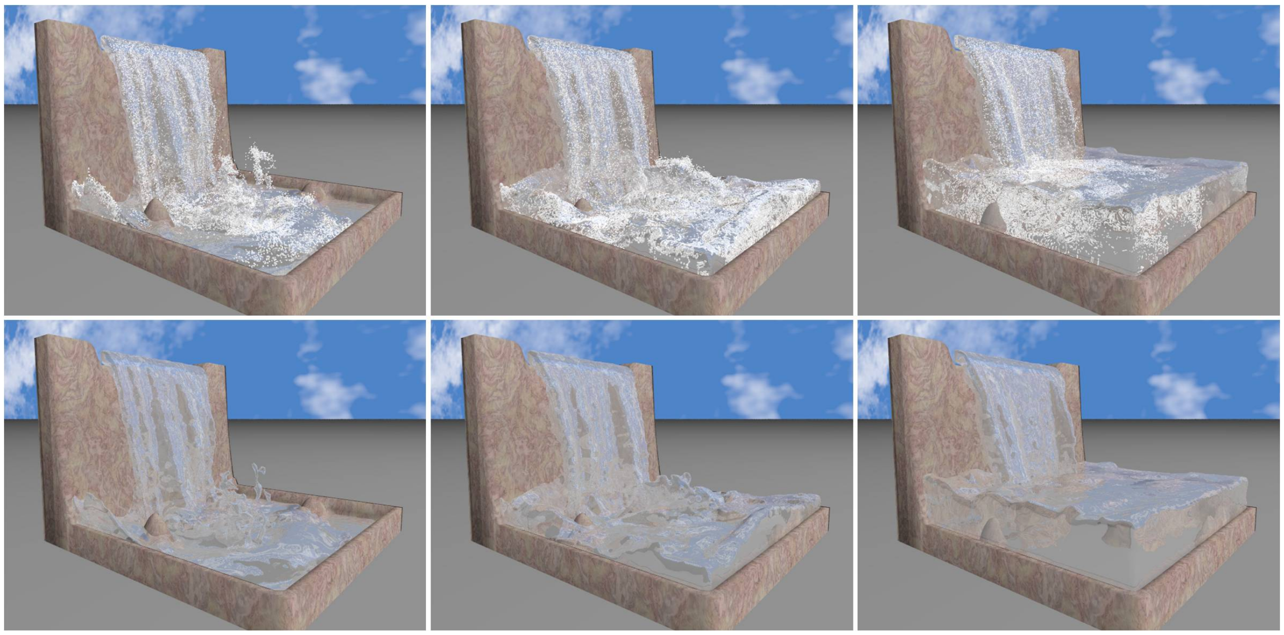

Figure 11: *Waterfall. Liquid stream is pouring down into a tank of water, with generated spray, bubble, and foam (refer to the first row), and the second row shows the original simulation result of FLIP method.*

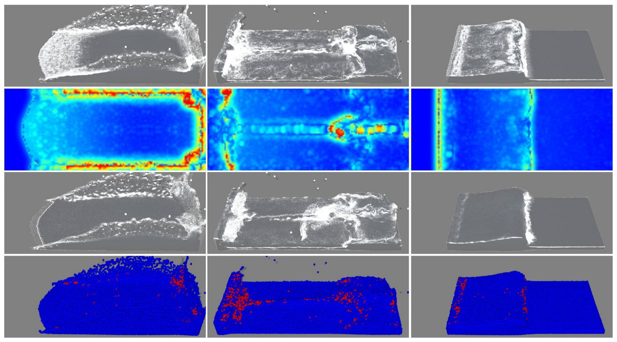

Figure 12: *The comparison of the results from our paper (top two rows) and that from [Ihmsen et al. 2012] (bottom two rows).*

improvement in terms of efficiency, primarily benefitting from our image-space-based GPU algorithmic architecture. However, when the method in [Ihmsen et al. 2012] is implemented on GPU, its diffuse particle generation and simulation costs around 30ms for a scenario involving 100k fluid particles.

The main **limitation** of our method is that we use a 2D depth buffer to estimate the 3D fluid surface information, when fluid occlusion happens, the depth buffer is not accurate, and thus our model cannot generate diffuse materials around the occluded fluid surface.

8 Conclusion and Future Work

In this paper, we have detailed a novel integrated framework for diffuse material animation and its detail enhancement by introducing a new analysis-and-simulation approach. The technical essence of our novel approach is the unification of time-space analysis in image domain and 3D physical simulation, built upon a CUDA-centric computational framework. The key innovation is that, the geometry and physics based criteria, together with time-space integrated strategy, can be tightly coupled into a logistic regression model. Our method showcases the detail enhancements of complex fluid interaction phenomena, and affords detail-preserving interaction while guaranteeing the high efficiency even for scenes with $212.9k$ liquid particles and $40.8k$ diffuse particles. The involved data-driven analysis method and diffuse material simulation can be integrated into any grid-based method with very little extra workload, because our approach only depends on the captured depth buffer and certain local information during simulation.

Our ongoing efforts include directly extending our time-space analysis model to process video data recording real fluid, which should offer more realistic diffuse material and more detail-informative free surfaces with better visual effects. Our novel approach is also readily available to be integrated with other available VR techniques to handle other types of visual data. In addition, the two-way coupling simulation between diffuse materials and liquid also deserves our further investigation.

Acknowledgement: This research is supported by National Natural Science Foundation of China (No.61190120, No.61190125, No.61190124, 61300067, and 61532002), and NSF (IIS-0949467, IIS-1047715, and IIS-1049448). We also thank the anonymous reviewers for their constructive critiques.

References

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

ANDO, R., THUREY, N., AND TSURUNO, R. 2012. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Trans. Visualization and Computer Graphics 18*, 8, 1202–1214.

BAGAR, F., SCHERZER, D., AND WIMMER, M. 2010. A layered particle-based fluid model for real-time rendering of water. *Comput. Graph. Forum 29*, 4, 1383–1389.

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

CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Probabilistic deformable surface tracking from multiple videos. In *ECCV*. 326–339.

CARMI, A., SEPTIER, F., AND GODSILL, S. J. 2012. The gaussian mixture MCMC particle algorithm for dynamic cluster tracking. *Automatica 48*, 10, 2454 – 2467.

CIOCCA, G., CUSANO, C., AND SCHETTINI, R. 2013. Image orientation detection using lbp-based features and logistic regression. *Multimedia Tools and Applications*, 1–22.

CORNELIS, J., IHMSEN, M., PEER, A., AND TESCHNER, M. 2014. Iisph-flip for incompressible fluids. *Computer Graphics Forum 33*, 2, 255–262.

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

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 23–30.

GERSZEWSKI, D., AND BARGTEIL, A. W. 2013. Physics-based animation of large-scale splashing liquids. *ACM Trans. Graph. 32*, 6, 185:1–185:6.

GOSWAMI, P., SCHLEGEL, P., SOLENTHALER, B., AND PAJAROLA, R. 2010. Interactive sph simulation and rendering on the GPU. In *Proc. of SCA'10*, 55–64.

HAMMOND, D. K., VANDERGHEYNST, P., AND GRIBONVAL, R. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis 30*, 129–150.

HE, K., SUN, J., AND TANG, X. 2013. Guided image filtering. *IEEE Trans. PAMI 35*, 6 (June), 1397–1409.

HONG, J.-M., AND KIM, C.-H. 2003. Animation of bubbles in liquid. *Comput. Graph. Forum 22*, 3, 253–262.

HONG, J.-M., LEE, H.-Y., YOON, J.-C., AND KIM, C.-H. 2008. Bubbles alive. In *ACM SIGGRAPH*, 48:1–48:4.

IHMSEN, M., BADER, J., AKINCI, G., AND TESCHNER, M. 2011. Animation of air bubbles with sph. In *GRAPP*, 225–234.

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

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH fluids in computer graphics. *Eurographics 2014-State of the Art Reports*, 21–42.

KHAN, Z. H., AND GU, I.-H. 2011. Bayesian online learning on riemannian manifolds using a dual model with applications to video object tracking. In *ICCV Workshops*, 1402–1409.

KIM, J., CHA, D., CHANG, B., KOO, B., AND IHM, I. 2006. Practical animation of turbulent splashing water. In *Proc. of SCA'06*, 335–344.

KROG, Ø. E., AND ELSTER, A. C. 2012. Fast gpu-based fluid simulations using sph. In *Applied Parallel and Scientific Computing*, vol. 7134. 98–109.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH*, 457–462.

LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG 14*, 4 (July), 797–804.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2009. Simulation of two-phase flow with sub-scale droplet and bubble effects. *Comput. Graph. Forum 28*, 2, 229–238.

MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1, 543–574.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, 154–159.

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. In *Proc. of SCA'05*, 237–244.

NARANG, S. K., AND ORTEGA, A. 2013. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE TSP 61*, 19, 4673–4685.

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

PATKAR, S., AANJANEYA, M., KARPMAN, D., AND FEDKIW, R. 2013. A hybrid lagrangian-eulerian formulation for bubble generation and dynamics. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, 105–114.

SELINO, A., AND JONES, M. 2013. Large and small eddies matter: Animating trees in wind using coarse fluid simulation and synthetic turbulence. *Computer Graphics Forum 32*, 1, 75–84.

SOLENTHALER, B., AND PAJAROLA, R. 2008. Density contrast sph interfaces. In *Proc. of SCA'06*, 211–218.

TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., AND UEKI, H. 2003. Realistic animation of fluid with splash and foam. *Comput. Graph. Forum 22*, 3, 391–400.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 839–846.

VAN DER LAAN, W. J., GREEN, S., AND SAINZ, M. 2009. Screen space fluid rendering with curvature flow. In *Proc. of I3D*, 91–98.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via l0 gradient minimization. *ACM Trans. Graph. 30*, 6 (Dec.), 174:1–174:12.

YANG, L., LI, S., HAO, A., AND QIN, H. 2014. Hybrid particle-grid modeling for multi-scale droplet/spray simulation. *Computer Graphics Forum 33*, 7, 199–208.

ZHENG, W., YONG, J.-H., AND PAUL, J.-C. 2006. Simulation of bubbles. In *Proc. of SCA'06*, 325–333.

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