Novel Fluid Detail Enhancement based on Multi-Layer Depth Regression Analysis and FLIP Fluid Simulation

Yuxing Qiu[†], Lipeng Yang[†], Shuai Li[†,*], Qing Xia[†], Hong Qin[‡], and Aimin Hao[†]

Abstract

In this paper, we propose a novel integrated method for effective modeling and realistic enhancement of scale-sensitive fluid simulation details. The core of our method is the organic of multi-layer depth image regression analysis and FLIP fluid simulation, of which, the regression analysis induces the criterion where the fluid details should be produced. First, we capture the depth buffer of the fluid surface dynamically from the top of scene. Second, we employ depth peeling technique to decompose the target fluid volume into multiple depth layers and conduct time-space analysis over surface layers. Third, we propose a logistic regression based model to rigorously pinpoint the

*Corresponding author: lishuai@buaa.edu.cn.

[†]Y. Qiu, L. Yang, S., Q. Xia, and A. Hao are with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University.

[‡]H. Qin are with the Department of Computer Science, Stony Brook University.

complex interacting regions, wherein multiple detail-relevant factors are taken into account based on the captured multiple depth layers. Finally, details are enhanced by animating extra diffuse materials and augmenting the air-fluid mixing phenomenon. It is evident that, with depth peeling technology, we can afford rigorous analysis not only across surface layers at different fluid depth, but along the depth direction as well. After integrating the analysis results from these two sources, we are capable of performing detail enhancement both on the fluid surface and inside the fluid to obtain a great visual effect, even when large occlusion exists. Directly benefiting from the flexibility of image-space-dominant processing, our unified framework can be entirely implemented on GPUs, and thus achieves interactive performance. For various fluid phenomena with different diffuse materials (e.g., spray, foam, and bubble), comprehensive experiments and evaluations have demonstrated its superiority in high-fidelity fluid detail enhancement and its interaction with surrounding environment.

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

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 dealing with small-scale splash, spray, and other diffuse materials [1, 2] for large-scale fluid simulations, 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 often-times resort to various space-grid refinement or denser particle sampling schemes in physical simulation. In addition, 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 [3, 4, 5] in a physically-plausible fashion to increase visual fidelity. Considering the unavoidable improve of computational expenses, adaptive sampling methods are more favourable, such as octree-grid [6], adaptive SPH [7], etc. Meanwhile, parallel methods on multi-core CPUs or GPU are also developed in order to reduce the ever-increasing computational cost [8, 9], while still improving visual quality.

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 propose to utilize multi-level depth buffers captured during simulation as an indicator of the dynamic fluid status to analyze the region-variant necessity of detail enhancement. 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 trade-off.
- We design a multi-layer surface information extraction method based on depth peeling to enhance the fluid simulation details over (multi-layered) surface regions.
- We develop a seeding method for diffuse particle generation, which only relies on the depth buffers and analysis results, making it applicable for all particle-based simulation methods.
- Without applying depth peeling technique, the involved data-driven analysis method and diffuse material simulation can be integrated into any grid-based method with very little extra workload with some specific surface extraction algorithm for grid-based fluid.

- We design specific GPU-based algorithms to implement our entire simulation framework in parallel.

Related Work

In recent years, there have been many methods developed to simulate fluids, including Smoothed Particle Hydrodynamics (SPH) method [10, 11, 12, 13], Fluid Implicit Particle (FLIP) method [14], and Level Set methods [15, 16], 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 [14], and then was extended to many fields, such as splashing water simulation [17], preservation of fluid sheet [18], fluid-solid coupling applications [19], combining with particle method [20], multi-scale droplet/spray simulation [21], etc. Lately, [18], [22] and [20] 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, data-specific analysis, and depth peeling technology separately.

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 [3], markers escaped from surface [4, 1, 23], etc. To exploit the ignored velocity information, [5] 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 [24], regional level set [25], MultiFLIP [26], two-continua approach [27], etc. [28] proposes a hybrid method for bubbles, using level set 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, [29, 30, 31] achieve detail enhancement based on SPH model. Although these methods can obtain improved visual results, however, they still have problems when certain qualifications required in their papers cannot be guaranteed. To ameliorate, [32] simulates the bubbles and liquids separately by differentiating the regions where air is likely to be trapped with a velocity-based heuristic criterion. [33] propose a two-way coupled simulation framework that uses the particle level set method to model dense liquid volumes and a SPH model to simulate diffuse regions such as sprays. [34] 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, [2] 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. [35] proposes a similar method for grid based simulator. Although having avoided expensive inter-particle computation, such schemes heavily rely on the proper detection of surface particles and the curvature calculation over surface particles.

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 produce plausible results in image smoothing and detail enhancement. For instance, in order to achieve data-specific wavelet, [36] proposes a method for the construction of wavelet transforms, according to the functions defined on the vertices of an arbitrary weighted graph. [37] constructs the data-specific wavelet by specifically designing wavelet filters based on the spectral decomposition of the graph, and then necessary and sufficient conditions for a two-channel graph filter bank on bipartite graphs are stated, which can achieve aliasing-cancellation and perfect reconstruction. In addition, [38] designs an optimization framework via L_0 gradient minimization, wherein they can sharpen major edges by increasing the steepness of transition while eliminating a manageable degree of low-amplitude structures at the same time. However, the computational costs of these methods are expensive, and they are not suitable for interactive VR applications. Guided image filtering is derived from a local linear model according to the texture pattern of guidance image. Compared with the widely used bilateral filter [39], guided filter naturally is a generic linear-time algorithm, which has a better trade-off between the accuracy and efficiency. Thus, considering both the structure-preserving

effect and the low computational cost demands, we introduce the guided image filter [40] into our method.

Depth Peeling Techniques. According to [41], depth peeling is a fragment-level depth sorting technique described by Mammen using Virtual Pixel Maps [42] and by Diefenbach using a dual depth buffer [43]. This technique is widely used in transparent surfaces rendering domain, and each unique depth in the scene is extracted into layers and composite in depth-sorted order, to produce a correctly-blended final image. With depth peeling, the rendering order-dependent problem of non-refractive transparent surfaces can be solved[41]. There exist some applications and improvements of depth peeling techniques. For instance, [44] proposes a new hardware-based ray casting algorithm for the depth peeling of unstructured meshes consisting of tetrahedral cells. In this way, they can capture the rays when they re-enter the mesh, without introducing imaginary cells to fill the intractable space caused by mesh non-convexities. Depth peeling techniques have also been extended to texture-based volumetric iso-surfaces rendering [45], in which the authors originate the term *volumetric depth peeling*. In this way, the full range of volumetric effects can be achieved. Inspired by this, [46] utilizes volumetric depth peeling for medical image visualization, which is flexible enough to handle multiple region occlusion and object's self-occlusion, and requires no pre-segmentation over the dataset. Based on depth peeling method, the joint surface occlusion problem has been solved, which has been successfully used in urology and visual arthroscopic studies [46]. However, the classical depth peeling algorithm has performance bottleneck for large and complex scenes, to improve it, [47] exploits multiple render targets

as bucket array in pixel-wise way. With the use of bucket sort and the efficient schemes to reduce collisions in the same bucket, [47] achieves 32 times speedup in depth peeling while guaranteeing faithful visual results, especially for complex scenes.

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 [48, 49], logistic regression [50], Gaussian mixture model [51], 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 or not.

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*.

Depth Peeling. Peel the multi-layer fluid surfaces according to the obtained depth buffer, and capture the depth of each layer for the subsequent analysis.

Space Analysis. Analyze the 2D depth buffer spatially with data-specific guided filter to obtain the geometric features within each layer of the near-surface.

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

Level Analysis. Conduct similar regression analysis over multi-layered depth maps to induce the criteria along depth direction.

Final Criteria. Integrate the regression results from the aforementioned two sources in a weighted way in order to obtain the final scale-sensitive criteria for detail enhancement over multi-layered surface regions.

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.

Brief FLIP Review

Since our model is based on the adaptive FLIP model [18], 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:

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

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

Here ρ is the density, $\mathbf{u}$ is the velocity, p is the pressure, μ is the kinematic viscosity, and $\mathbf{f}$ is the external force ∇ means the differential operator of vector.

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 [14], 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.

Time-space Analysis

Depth Image Sequence Capture

As shown in Fig. 2, the depth buffer is captured at the end of each regular FLIP update cycle. Similar to [52] (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 trade-off of 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 **Experiments, Evaluations, and Discussions**.

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 [40] to perform detail detection. Although some simple linear translation-invariant filters (such as the mean, Gaussian, Laplacian and Sobel filters) have been widely used in image processing, they are weak in making the smoothed image preserve original structures and edges. Utilizing guidance image instead of a fixed filter, we can take into account more data-specific information to make the smoothed image preserve structures and edges. In our application, we tend to extract the potential detail features from the differences of smoothed multi-level depth buffers and original ones. According to this, the edges of depth buffer need to be consistently persisted. Some sharp edges, which might represent the overlaps between different fluid flows and fluid pools, should not be considered as regions needing significant enhancement. Unlike bilateral filter, guided filter has good edge-preserving properties without suffering from gradient reversal artifacts. Besides, efficiency is also an important factor for us to choose guided filter, because the filtering output of guided filter is locally a linear transform of the guidance image. In sharp contrast, other content-specific filters usually utilize some optimizations to pursue better quality, and this oftentimes comes with the price of expensive computational time. Considering our eager expectation of simplicity and efficiency, we finally choose the guided filter in our method.

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 same depth buffer as the guidance image I and the input image p. According to [40], we set ϵ with an empirical value 0.01 in all our experiments, which can produce satisfactory results.

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, $|w|$ 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-level 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. 6 and our supplementary video).

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 the 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

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 θ_j is the corresponding coefficient to be learned, and x_j represents all the factors being taken into consideration. In regression-based time-space analysis of certain surface layer, x_1, x_2 and x_3 are separately set to be the difference of the depth images, the curvature of the depth images and the difference of the velocity fields. As for the regression-based level analysis, which will be introduced in the following sections, we set x_1 and x_2 as the differences and curvatures of the cross-layer depth images, and set x_3 as the velocity gradient in depth direction.

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. In our experiments, we found that 0.01 is a good choice for r, because the decrease of r does not obviously accelerate the convergence speed anymore; however, increasing r can slightly contribute to the visual fidelity improvement. As a result, we finally choose to set r as 0.01 for all phenomena in this paper. 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 to adjust the weight of the logistic regression cost and inconsistency cost between adjacent frames. Considering that consistency is important for the visual fidelity of fluid simulation, we set μ with a relatively large weight as 5. In fact, it is difficult to quantitatively evaluate the visual fidelity for the absence of a specific standard. Thus, we just use the same value for all scenes to achieve the simplification, and the results are considered to be satisfactory enough. 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:

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

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

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. As for α_1 and α_2, they represent the step length of the gradient descent method, we set them according to the empirical values, and the resulted convergence performance and the time consumption are satisfied. It should be noted that, these values are also kept same for all scenes (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 **CUDA-based Implementation**. Fig. 2(d) shows our time-space analysis results as an example.

Depth Peeling Method

We have discussed how to induce the criteria for detail enhancement from depth buffer captured from a downward orthogonal camera. However, this depth buffer only captures the information of the fluid surface, which is apparently far from being useful, because the bubbles may appear underwater. In order to calculate the full-scale generation criteria for diffuse particles and equip our model with much stronger ability to handle more complex cases (e.g., plunging breaker), we employ depth peeling technology [41] to acquire multi-layer depth buffers inside fluid. That is, we can get n depth layers deeper from the fluid particles beneath the surface via n-pass rendering of this scene.

In practice, after capturing the depth image of current surface, we peel the top layer particles according to this depth image (it is used to locate surface particles), so that we can expose the subsurface of the fluid. After peeling, the method detailed in Section **Depth Image Sequence Capture** will be applied again on the new surface to capture the depth buffer. Then the guided filter based space analysis (Section **Guided Filter based Space Analysis**) and regression based time-space analysis (Section **Regression-based Time-space Analysis**) are employed to conduct analysis across the surface layers, so that we can also obtain the velocity field, as well as the regression analysis result of this layer. Fig. 3(c-f) illustrate the time-space analysis results of the different layers obtained from depth peeling over surface regions. In this scene, two violent water streams on the left side are injected into a tank from underwater, while another two are spouted above the water on the right. We choose to use 5 layers to conduct analysis as an example, however, this method does not have a limitation in the number of depth layers, which is decided by the number of passes of depth peeling process. If we only take the surface information into account, the streams inside the fluid will be ignored and no diffuse particles will be generated for them. As shown in Fig. 3(b), it looks unreal and should have been avoided. It shows that different layer holds unique physical and geometric information at different fluid depth, which makes depth peeling step both meaningful and necessary. As we can see in this figure, the differences among the velocity fields of different layers are especially remarkable, which decide the imparities of the time-space analysis results to a large extent. In Section **Experiments, Evaluations, and Discussions**, we will conduct comparisons between the detail enhancement results with

and without depth peeling.

The tradeoff between the computational cost and analysis quality of depth peeling is controlled by two parameters: the number of peeling passes, and the range of height for one-time peeling, which is represented by a distance threshold λ from underwater particles to fluid surface. The more the peeling passes, the deeper we can reach underwater, and the more computational cost will be needed. On the other hand, the height range for one-time peeling can influence the quality of detail enhancement, because the depth and velocity field extraction results of current surface are accurate and make the diffuse particle generation analysis more reliable only at certain positions. However, the probability of detail enhancement around the two adjacent layers is calculated via linearly interpolating the two-layer regression results, which may be error-prone. Our parameter selection strategy will be discussed in Section **Experiments, Evaluations, and Discussions**.

Regression-based Level Analysis

During the several rounds of depth peeling, we can gradually extract the depth information as well as the velocity field of deeper layers under the fluid surface, which comprise a depth buffer sequence and a velocity sequence along the depth direction. For more thorough analysis, we propose regression-based level analysis, using the two sequences documented above to induce the criteria for detail enhancement along the depth direction. It should be noted that, the analysis introduced in Section **Regression-based Time-space Analysis** is

executed over (multi-layered) surfaces only.

By analyzing the curvatures of these sequential depth layers, we can leverage the variation rate of depth as geometrical criterion to focus on the locations with high-frequency depth changes along the depth direction. And we consider that, the faster the depth changes in vertical direction, the more likely the fluid is to interact with air. As the diffuse particles have higher probability to appear at fluid-air mixed positions, the second-order derivatives of the depth layers should be taken as one of the indicators to enhance details as Eq. 13, and the computed results are shown in Fig. 3(h).

$$g_i^{2d} = d_{i+1} - 2d_i + d_{i-1}, \tag{13}$$

where g_i^{2d} means the pixel-wise second-order derivative of depth layers, and d_i is the depth value of the i-th layer. Considering the depth gradient $(d_i - d_{i-1})$, we can use Eq. 13 to approximate the second-order derivative.

In addition, once we obtain the velocity fields of different fluid layers, we can calculate the differences of fluid velocity among adjacent layers, which serve as physical features as:

$$\mathbf{g_i^v} = \frac{\mathbf{v_{i+1}} - \mathbf{v_i}}{2} + \frac{\mathbf{v_i} - \mathbf{v_{i-1}}}{2} = \frac{\mathbf{v_{i+1}} - \mathbf{v_{i-1}}}{2}, \tag{14}$$

where $\mathbf{g_i^v}$ is the pixel-wise velocity gradient in depth field, and $\mathbf{v_i}$ is the velocity at this position in the i-th depth layer. In common sense, the bubbles are more likely to appear

around the locations where the fluid velocities in the neighboring layers are quite different. The results of velocity analysis in depth direction can be found in Fig. 3(i).

Both the geometric and the physical characteristics calculated above will be used to determine the potential detail regions. Similar to the method used in Section **Regression-based Time-space Analysis**, we employ the logistic regression model to estimate the probability of diffuse particles generation along the depth direction, with the aforementioned geometric and the physical analysis results as inputs. And the time-space analysis result along the depth direction is shown in Fig. 3(j), while Fig. 3(f) illustrates the time-space analysis results over surface regions only.

After level analysis, we can obtain the regression results both over surface regions and along the depth direction, and thus the final criterion for detail enhancement can be calculated by $y[i] = \omega_1 * y_1[i] + \omega_2 * y_2[i]$ for certain depth layer, wherein $y_1[i]$ is the regression result of time-space analysis on surface, $y_2[i]$ represents the result of level analysis in the depth direction, and i is the layer index. The weighted combination of $y_1[i]$ and $y_2[i]$ gives rise to the final regression results, and the weight coefficients ω_1 and ω_2 satisfy $\omega_1+\omega_2 = 1.0$. In all experiments, ω_1 is set to be 0.7 and ω_2 is set to be 0.3. The probability of the diffuse particles generation between two adjacent depth layers is calculated via linearly interpolating the criteria over those two layers. Fig 3(g) shows the final analysis results for seeding diffuse particles at certain depth layer.

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 [2], but we have modified the method to accommodate FLIP framework and GPU acceleration.

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 in a specific spatial range around the projected grid node. We utilize the depth image to generate diffuse particles, rather than relying on the particle distribution [2], 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 $t_{life} \leq 0$, we remove the particle. The diffuse particle classification will be described in Section **Advection of Diffuse Particles**. 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.

Advection of Diffuse Particles

To advect diffuse particles in a more specific way, we classify them into spray, foam, and bubble according to their relative position to the fluid surface. Spray, foam and bubble particles corresponding have different advection formulations, which will be introduced in the following paragraphs. 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. And the diffuse particles floating around the surface are classified as foam particle. What's more, the diffuse particles under the fluid surface are regarded as bubble particles. The classification task can be accomplished by computing the liquid particle number in the neighboring volume. Fig. 4 illustrates the criterion and the result of our 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 $\mathbf{g}$.

$$\mathbf{f}_{dp} = k_d(\mathbf{v}_{dp} - \mathbf{v}_f) - k_b\mathbf{g}, \tag{15}$$

where k_d, k_b separately correspond to the coefficients of drag force and buoyancy force used to advect bubble particles. The buoyancy force on a submerged body points to the opposite direction to gravity and is equal to $\rho V\mathbf{g}$, where ρ is the density of the fluid, V measures the volume of the underwater bubbles and $\mathbf{g}$ means the gravity acceleration. Compared with our proposed formulation, k_b is exactly calculated by $\rho * V$. According to this, k_b should be a constant for all scenes mentioned in this paper, as we use bubble particles with the same size and all diffuse particles are generated for water scenes. If the fluid property is changed (especially the density of fluid),the value of k_b also needs to be adjusted. And as for k_d, we take it as a constant to control drag effects. And k_d is chosen as 1 in order to make air bubbles immediately dragged into the fluid flow direction to enhance the visual fidelity according to [2], which are also kept the same for all scenes. The subscript dp is always associated with a variable of diffuse particle. Here, $\mathbf{v}_{dp}$ and $\mathbf{v}_f$ separately represent the velocity of diffuse particles and fluid particles.

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.

CUDA-based Implementation

Since FLIP based simulations have been implemented on GPU efficiently [21], 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.

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 8-12 of Algorithm 1. We need two kernel functions for line 10 and line 12, 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_1 and verify whether the convergence condition is satisfied. In the second kernel function, we update y_1 for each pixel in a parallel way. When the computation is completed,

the analysis result y_1 is stored in the GPU array to be used in the following steps.

Depth Peeling and Level Analysis. We employ a CUDA kernel to compute the coordinates of fluid surface according to the depth map, and invoke another CUDA kernel to compute the distances from each particle to the fluid surface, and then decide whether the particle needs to be peeled or not based on a distance threshold λ.

To conduct level analysis, we first need to calculate the second-order derivatives of depth maps and the differences of velocity field in the depth direction, therefore, we invoke two CUDA kernels at each pixel to compute these two items separately. As shown on the line 18-22 of Algorithm 1, we employ the two kernels mentioned in *Regression-based Time-space Analysis* to compute θ_2 and y_2. And a CUDA thread for weighted combination of y_1 and y_2 is executed at last.

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, 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, and further set the velocities and life time respectively. The generated diffuse particles are dynamically inserted into the data

structure proposed above, 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 particles (by using Eq.15). 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.

Experiments, Evaluations, and Discussions

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. In addition, we have re-rendered our experimental result using POV-Ray. Considering that the rendering algorithm does not truly dominate our subject, we finally chose the same particle-based rendering method to reduce the complexity as much as possible, and the resulted visual results are satisfactory. However, more complex rendering algorithm is also significant for better visual fidelity, for example,

Algorithm 1: Analysis-driven Detail Enhancement.

n in the pass number of depth peeling
for $i = 1$ *to* n **do**
    capture the depth buffer d directly from scene.
    smooth d with guided filter method.
    compute the difference $dif[i]$.
    compute the velocity field $\mathbf{v}[i]$.
    initialize $y_1[i]$, $\theta_1[i]$;
    while *not converged* **do**
        while *not converged* **do**
            update $\theta_1[i]$ with Eq. 11
        while *not converged* **do**
            update $y_1[i]$ with Eq. 12
    peel surface particles according to $d[i]$ and distance threshold λ.
for $i = 2$ *to* n **do**
    compute the second-order derivative $g_i^{2d}[i]$ in the depth direction.
    compute the difference of velocity field $\mathbf{g_i^v}[i]$ in the depth direction.
    initialize $y_2[i]$, $\theta_2[i]$;
    while *not converged* **do**
        while *not converged* **do**
            update $\theta_2[i]$ with Eq. 11
        while *not converged* **do**
            update $y_2[i]$ with Eq. 12
    compute a weighted combination of $y_1[i]$ and $y_2[i]$ to obtain $y[i]$.
base on linear interpolation, compute $yAll$ according to y.
seed diffuse particles at the projected grid node according to d and $yAll$.
partition the array by thrust library.
advect all the diffuse particles with positive t_{life}.

methods introduced by [53] and [54] can be used to obtain the more faithful visual fidelity. 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. 5 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. 6, our result reveals scale-aware features effectively.

Table 1 documents the statistics for the average testing time (in milliseconds) of each simulation cycle. We record the number of FLIP particles (#FP) and the maximum number of diffuse particles (#DPM) for each case. To be clear, (AN) and (DP) are the time costs for time-space analysis and diffuse particle generation and advection separately, (GR) means the grid resolution while (DBR) refers to *depth buffer resolution*, and (*Other*) is the time cost for deformable solid simulation and fluid-solid coupling. In comparison with the computational time of FLIP simulation (refer to (FLIP) column), the overhead of our time-

space analysis and diffuse material simulation is small, while improving the visual effects significantly. The depth peeling technique is introduced into the simulation of dam-break scene and pouring scene. Some parameters and time statistics are shown in the subsequent columns, wherein (#D) means the passes of depth peeling (i.e., the number of depth layers) and (λ) represents the distance threshold for one-time peeling. The time cost of time-space analysis for all layers (AND) is approximately equal to (#D) the times of the analysis cost for one layer (AN), that is, the time cost of time-space analysis will increase in nearly linear way with the number increasing of depth layers. And level analysis (ANL) costs similar computational time as (AN), no matter how many depth layers there are. (DPP) refers to the time used for diffuse particle generation after depth peeling. Considering that the diffuse particles generated for each grid in one frame is limited, there is no definite multiple relationship between (DP) and (DPP), as shown in the last column of Table 1. In order to show how significantly the computational cost will increase when the layer number increases, we separately document the time cost of time-space analysis(AND), level analysis(ANL), and diffuse particle generation after depth peeling(DPP) in Table 2. And the time cost change caused by varying the distance threshold for one-time peeling(λ) is also considered in this table. As we can see from the last line of Table 2, λ does not significantly influence the computation cost. And Fig. 14 documents the parameter values used in our experiments.

To further explore how significantly the parameter settings can influence the final visual results, we conduct some experiments under different parameter settings, and the results are shown in Fig. 14. Considering that the visual effects of detail enhancement are determined

by the regression results, we only record and compare the output of logistic regression model when it adopts different parameter values, taking the logistic regression results of the first depth layer of frame 250 in pouring scene as example. Using the parameter listed in the top right corner of Fig. 14 as basic setting, we can obtain the regression results as Fig. 14(a). When setting r as 0.001 and 0.1 and keeping other parameter unchanged, we can obtain Fig. 14(b1) and Fig. 14(c1) separately. And the difference between (b1) and (a) is shown in Fig. 14(b2), the same numbering rule is used for other sub-figures. It is remarkable that, only little difference can be observed when we change the value of parameter r, and the similar thing also happens when we change other parameter values. This phenomenon can reflect that, our method are insensitive to parameter changes. Accordingly, to keep simplicity, we use the same parameters' setting in all our experiments, as the basic parameter settings shown in Fig 14.

Fig. 4 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. 7. 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 cannot 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. 8 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 [21], our method ignores the smoke-like sprays, nonetheless, we are able to simulate spray, bubble, and foam in a unified framework, which indicates that our method is more flexible and has a stronger capability for more complex scenarios.

Fig. 9 demonstrates a pouring scene, where two violent above-water-level streams are spouted into the water tank on the right and another two streams on the left are injected from underwater. As the streams pouring down over the water from the right, numerous spray particles are generated to enhance the visual details. Furthermore, when streams flowing into the water volume, it is expected that plentiful bubbles should appear. Comparing with the depth peeling involved result (illustrated in the first row), we can clearly see that, the simulation results without depth peeling cannot faithfully exhibit impetuous torrent of underwater (refer to the second row), which definitely looks less realistic. The same things happen in Fig. 10, where the sea waves roll over surfaces again and again. Without depth peeling, we cannot deal with the details under the overlapping sea waves, and the bubbles

would not appear under the sea surface even if the undercurrents have great velocities (refer to the first column of Fig. 10).

In order to evaluate how the parameters of depth peeling impact the simulation results, we conduct some experiments with sharp contrast in Fig. 11. Considering the water-pouring scene as an example, the first line of Fig. 11 has a varying distance threshold ($\lambda = d_p, 2d_p, 3d_p, 4d_p$ and $5d_p$) for each pass of depth peeling. As we can see, when the number of peeling layer is fixed, particles (after each depth peeling) with small distance threshold may cause incomplete diffuse materials below the water level (refer to Fig. 11(a)), while the results would not improve gradually as the distance threshold value growing constantly and even over-fitting phenomena may appear (see Fig. 11(e)). If we make the distance threshold λ be constant and set the number of depth layers $\#D$ to range from 1 to 5, the second line of Fig. 11 can be obtained. As shown in Fig. 11(f-g), too few layers may influence the experimental results, however, too many layers are meaningless for a certain scene, since no obvious performance improvement could be recognized from Fig. 11(h-j) while the computational cost increases significantly.

Fig. 12 shows the comparison between our method and that in [2]. Comparing with [2], our method defines completely different criteria to facilitate diffuse material generation, and can achieve competitive visual effects. It may be noted that, [2] 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

benefiting from our time-space analysis, the results evolve continuously during the simulation (please refer to our supplementary video). Moreover, our method has a significant improvement in terms of efficiency, primarily benefiting from our image-space-based GPU algorithmic architecture. However, when the method in [2] is implemented on GPU, its diffuse particle generation and simulation costs around 30ms for a scenario involving 100k fluid particles.

According to Fig. 9 and Fig. 10, it is easy to observe that, when fluid occlusion happens, **without depth peeling** the depth buffer can only capture the surface information, and thus cannot generate diffuse materials around the occluded fluid surface and inside water volume as well. However, after introducing **depth peeling technology**, we can obtain the depth and velocity information as deep as we desire inside the fluid volume through flexible parameter control, which affords us a new way for underwater analysis and enables us to handle more scenes even with fluid occlusion and overlapping phenomena. On the other hand, even though using depth peeling could improve the visual effect for detail enhancement to a great extent, it necessarily increases the computational cost for peeling and analysis.

In order to analyze how the adopted gird size can influence the visual effect, as shown in Fig. 13, we take frame 350 of the pouring scene as example to show the experiment results under different grid sizes. Suppose that the total size of this scene is represented as $1.5\times1\times1$, and the FLIP grid resolution is calculated using the total size divide by grid size. As shown in Fig. 13, the grid size can significantly influence the FLIP simulation results. Considering that our diffuse particle generalization method is actually based on the FLIP

fluid simulation results, the final visual effects will be affected by the grid size. However, as shown in Fig. 13, no matter what the grid size is, the fluid details can be enhanced properly with our model.

Conclusion and Future Work

In this paper, we have detailed a novel integrated framework for diffuse material animation and its visual 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 coupled into a logistic regression model. Moreover, the depth peeling method could improve the applicability and reality of visual detail enhancement beneath fluid surface and help us overcome difficulties when handling underwater scenes caused by fluid occlusion. 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 other fluid simulation grid-based method with very little extra workload, because our approach only depends on the captured depth buffer and certain local information during simulation. However, it is still difficult to apply depth peeling to grid-based method

unless particles are automatically combined with the grid like FLIP, while generalizing depth peeling scheme to particle-based fluid algorithm is simple.

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.

Acknowledgment: 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

[1] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG*, 14(4):797–804, July 2008.

[2] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner. Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput.*, 28(6-8):669–677, June 2012.

[3] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki.

Realistic animation of fluid with splash and foam. *Comput. Graph. Forum*, 22(3):391–400, 2003.

[4] J. Kim, D. Cha, B. Chang, B. Koo, and I. Ihm. Practical animation of turbulent splashing water. In *Proc. of SCA'06*, pages 335–344, 2006.

[5] V. Mihalef, D. Metaxas, and M. Sussman. Simulation of two-phase flow with sub-scale droplet and bubble effects. *Comput. Graph. Forum*, 28(2):229–238, 2009.

[6] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH*, pages 457–462, 2004.

[7] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):481–487, July 2007.

[8] P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola. Interactive sph simulation and rendering on the GPU. In *Proc. of SCA'10*, pages 55–64, 2010.

[9] Ø. E. Krog and A. C. Elster. Fast gpu-based fluid simulations using sph. In *Applied Parallel and Scientific Computing*, volume 7134, pages 98–109. 2012.

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

[11] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 154–159, 2003.

[12] Y. Lipeng, L. Shuai, H. Aimin, and Q. Hong. Realtime two-way coupling of meshless fluids and nonlinear FEM. *Computer Graphics Forum*, 31(7):2037–2046, 2012.

[13] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. SPH fluids in computer graphics. *Eurographics 2014-State of the Art Reports*, pages 21–42, 2014.

[14] Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005.

[15] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph.*, 21(3):736–744, July 2002.

[16] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 23–30, New York, NY, USA, 2001. ACM.

[17] D. Gerszewski and Adam W. Bargteil. Physics-based animation of large-scale splashing liquids. *ACM Trans. Graph.*, 32(6):185:1–185:6, 2013.

[18] R. Ando, Nils Thurey, and Reiji Tsuruno. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Trans. Visualization and Computer Graphics*, 18(8):1202–1214, 2012.

[19] A. Selino and M.D. Jones. Large and small eddies matter: Animating trees in wind using coarse fluid simulation and synthetic turbulence. *Computer Graphics Forum*, 32(1):75–84, 2013.

[20] J. Cornelis, M. Ihmsen, A. Peer, and M. Teschner. Iisph-flip for incompressible fluids. *Computer Graphics Forum*, 33(2):255–262, 2014.

[21] Y. Lipeng, L. Shuai, H. Aimin, and Q. Hong. Hybrid particle-grid modeling for multi-scale droplet/spray simulation. *Computer Graphics Forum*, 33(7):199–208, 2014.

[22] K. Um, S. Baek, and J. Han. Advanced hybrid particlegrid method with subgrid particle correction. *Computer Graphics Forum*, 33(7):209–218, 2014.

[23] J.-M. Hong, H.-Y. Lee, J.-C. Yoon, and C.-H. Kim. Bubbles alive. In *ACM SIGGRAPH*, pages 48:1–48:4, 2008.

[24] J.-M. Hong and C.-H. Kim. Animation of bubbles in liquid. *Comput. Graph. Forum*, 22(3):253–262, 2003.

[25] W. Zheng, J.-H. Yong, and J.-C. Paul. Simulation of bubbles. In *Proc. of SCA'06*, pages 325–333, 2006.

[26] L. Boyd and R. Bridson. Multiflip for energetic two-phase fluid simulation. *ACM Trans. Graph.*, 31(2):16:1–16:12, April 2012.

[27] M. B. Nielsen and O. Osterby. A two-continua approach to eulerian simulation of water spray. *ACM Trans. Graph.*, 32(4):67:1–67:10, July 2013.

[28] S. Patkar, M. Aanjaneya, D. Karpman, and R. Fedkiw. A hybrid lagrangian-eulerian formulation for bubble generation and dynamics. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 105–114, 2013.

[29] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *Proc. of SCA'05*, pages 237–244, 2005.

[30] B. Solenthaler and R. Pajarola. Density contrast sph interfaces. In *Proc. of SCA'06*, pages 211–218, 2008.

[31] N. Threy, F. Sadlo, S. Schirm, M. Mller-Fischer, and M. Gross. Real-time simulations of bubbles and foam within a shallow water framework. In *ACM Siggraph/eurographics Symposium on Computer Animation, SCA 2007, San Diego, California, Usa, August*, pages 191–198. 2007.

[32] M. Ihmsen, J. Bader, G. Akinci, and M. Teschner. Animation of air bubbles with sph. In *GRAPP*, pages 225–234, 2011.

[33] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.

[34] F. Bagar, D. Scherzer, and M. Wimmer. A layered particle-based fluid model for real-time rendering of water. *Comput. Graph. Forum*, 29(4):1383–1389, 2010.

[35] W. Changbo, Z. Qiang, K. Fanlong, and Q. Hong. Hybrid particle-grid fluid animation with enhanced details. *Vis. Comput.*, 29(9):937–947, 2013.

[36] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30:129–150, 2011.

[37] S. K. Narang and A. Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE TSP*, 61(19):4673–4685, 2013.

[38] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l0 gradient minimization. *ACM Trans. Graph.*, 30(6):174:1–174:12, December 2011.

[39] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, Jan 1998.

[40] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. PAMI*, 35(6):1397–1409, June 2013.

[41] C Everitt. Interactive order-independent transparency. *White Paper*, 7(7-8):491503, 2001.

[42] M. Abraham. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Computer Graphics and Applications*, 9(4):43–55, 1989.

[43] P. J. Diefenbach. Pipeline rendering: Interaction and realism through hardware-based multi-pass rendering. *Ircs Technical Reports*, 1996.

[44] F. Bernardon Fbio, A. Pagot Christian, L. D. Comba Joo, and T. Silva Cludio. Gpu-based tiled ray casting using depth peeling. *Journal of Graphics Gpu and Game Tools*, 11(4):1–16, 2006.

[45] Z. Nagy and R. Klein. Depth-peeling for texture-based volume rendering. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 429–, Washington, DC, USA, 2003. IEEE Computer Society.

[46] B. David, C. John, Fielding Julia, R., and Taylorii Russell, M. Volumetric depth peeling for medical image display. *Proceedings of SPIE - The International Society for Optical Engineering*, 6060:606004–606004–11, 2006.

[47] L. Fang, H. Mengcheng, L. Xuehui, and W. Enhua. Bucket depth peeling. In *International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2009, New Orleans, Louisiana, USA, August 3-7, 2009, Talks Proceedings*, pages 1–1, 2009.

[48] C. Cagniart, E. Boyer, and S. Ilic. Probabilistic deformable surface tracking from multiple videos. In *ECCV*, pages 326–339. 2010.

[49] Z. H. Khan and I.Y.-H. Gu. Bayesian online learning on riemannian manifolds using a dual model with applications to video object tracking. In *ICCV Workshops*, pages 1402–1409, 2011.

[50] G. Ciocca, C. Cusano, and R. Schettini. Image orientation detection using lbp-based features and logistic regression. *Multimedia Tools and Applications*, pages 1–22, 2013.

[51] A. Carmi, F. Septier, and S. J. Godsill. The gaussian mixture MCMC particle algorithm for dynamic cluster tracking. *Automatica*, 48(10):2454 – 2467, 2012.

[52] W. J. Van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. In *Proc. of I3D*, pages 91–98, 2009.

[53] W. J. V. Der L., S. Green, and M. A. Sainz. Screen space fluid rendering with curvature flow. 2009.

[54] F. Bagar, D. Scherzer, and M. Wimmer. A layered particle-based fluid model for real-time rendering of water. *Computer Graphics Forum*, 29(4):1383–1389, 2010.

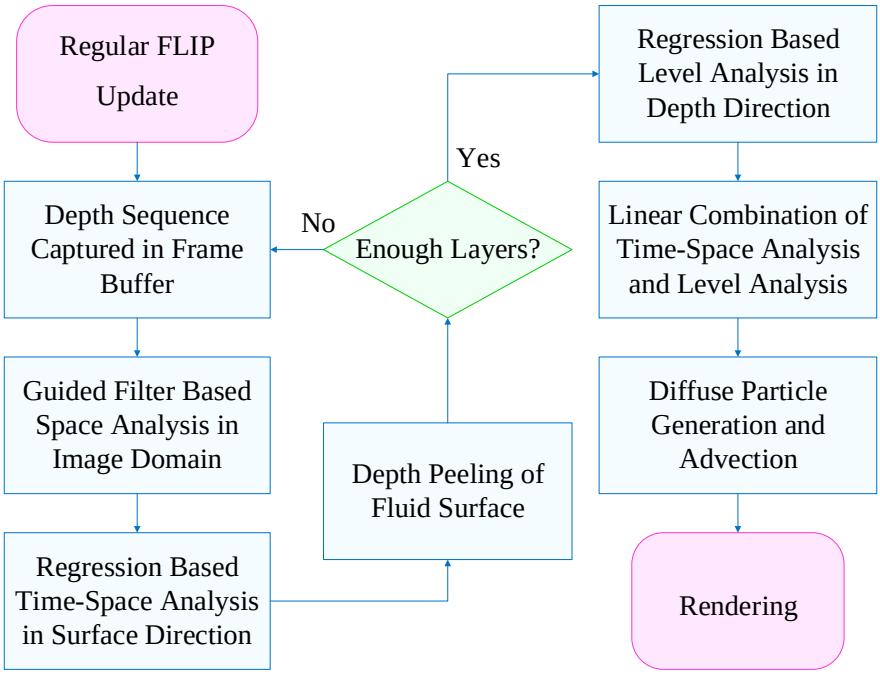


Figure 1: The pipeline of each simulation cycle.

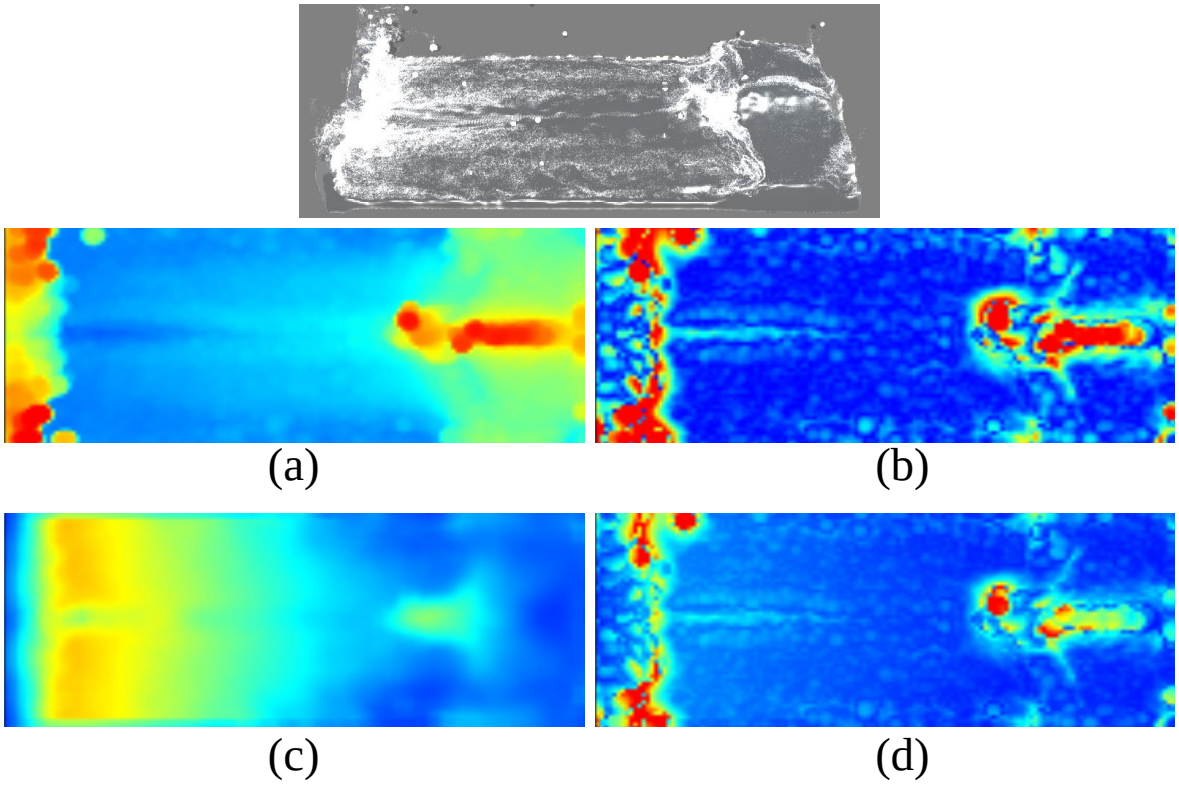


Figure 2: Time-space analysis results. Warm color indicates high value and vice versa. (a) The captured depth buffer, (b) The space analysis result, (c) Velocity field, (d) Time-space analysis result.

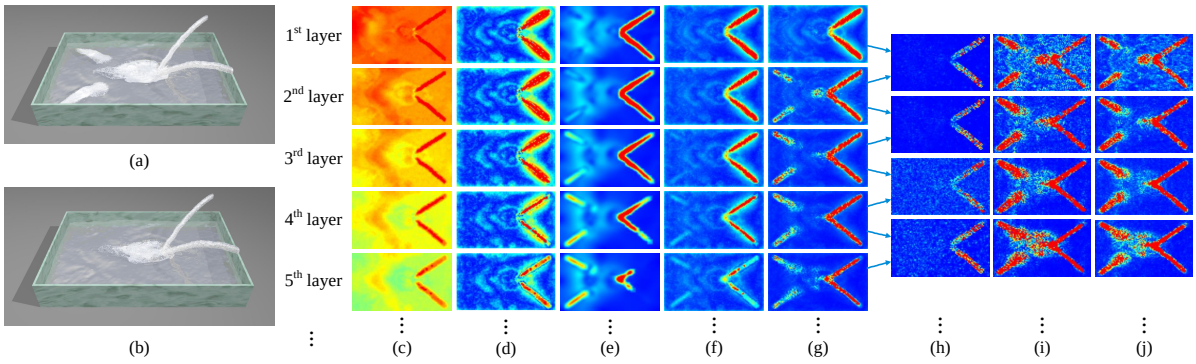


Figure 3: Time-space analysis results and level analysis results aided by depth peeling. Warm color indicates high value and vice versa. Fig. 3(c-f) are analyzed over surface regions, while (h-j) are processed along the depth direction. (a) The scene analyzed by the depth peeling technique, (b) Scene analyzed without depth peeling, (c) Captured multi-layer depth buffers, (d) Space analysis results, (e) Velocity fields, (f) Time-space analysis results, (g) The final analysis result used for seeding diffuse particles, (h) Space analysis result between neighboring layers, (i) Velocity analysis result, (j) Level analysis result.

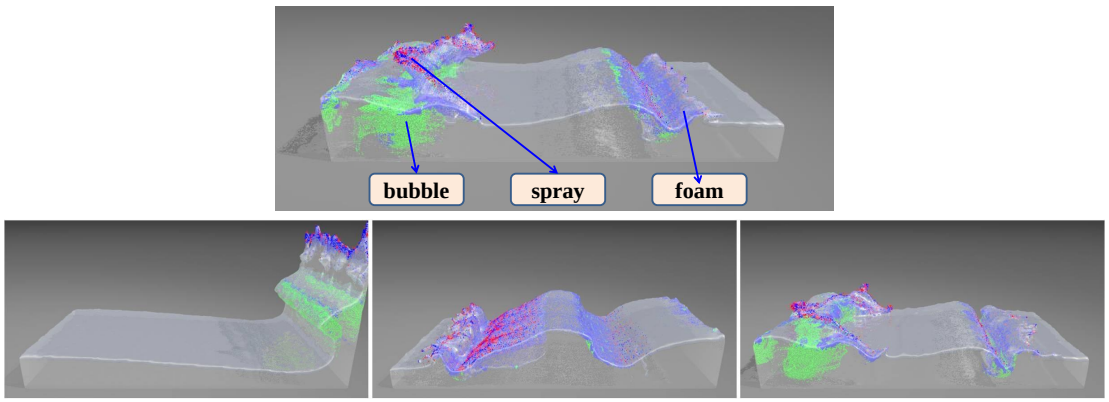
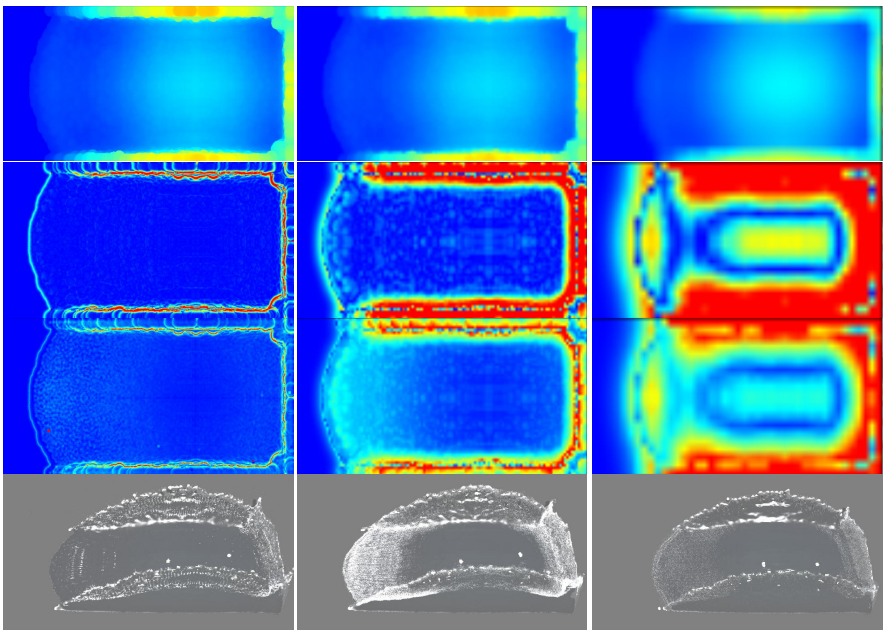


Figure 4: The classification of diffuse particles (red for spray particles, green for bubble particles, and blue for foam particles) using dam-breaking simulation as an example.



(a) (b) (c)

Figure 5: Results corresponding to different resolutions. Warm color indicates high value and vice versa. The resolution in simulation space is 140×60, and the depth buffer resolutions are: (a) 700×300, (b) 140×60, (c) 70×30.

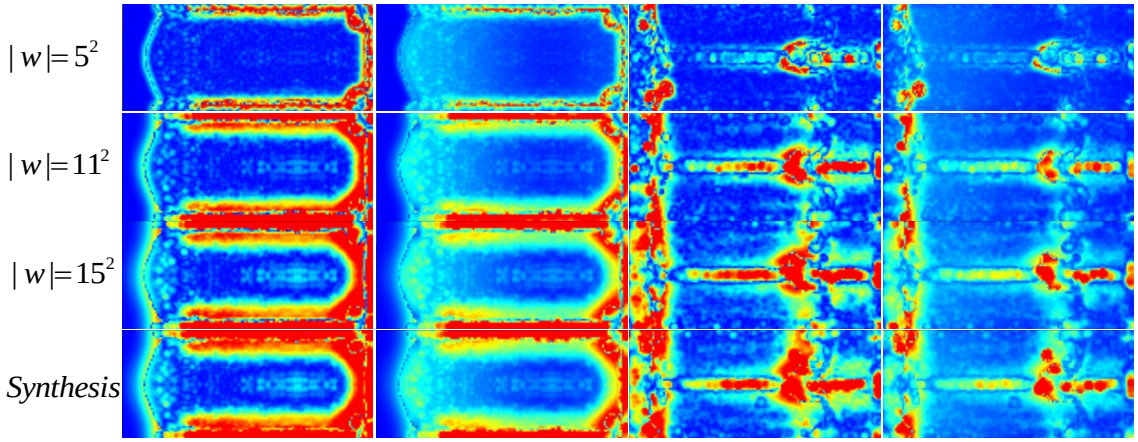


Figure 6: Our scale-aware analysis results. Warm color indicates high value and vice versa.

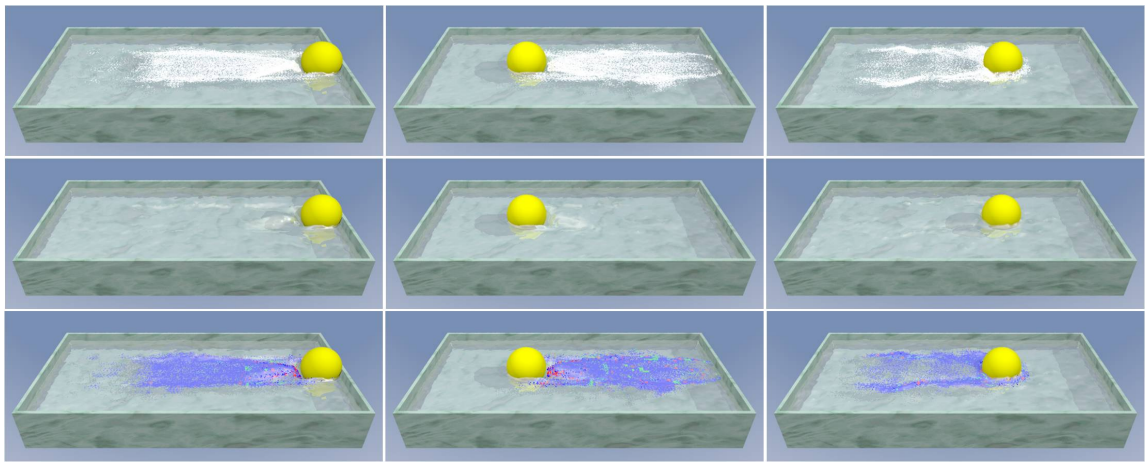


Figure 7: The interaction effects between solid and water. The first row shows simulation result with diffuse materials, the second row shows the original FLIP simulation result, and the third row shows the classification of diffuse materials.

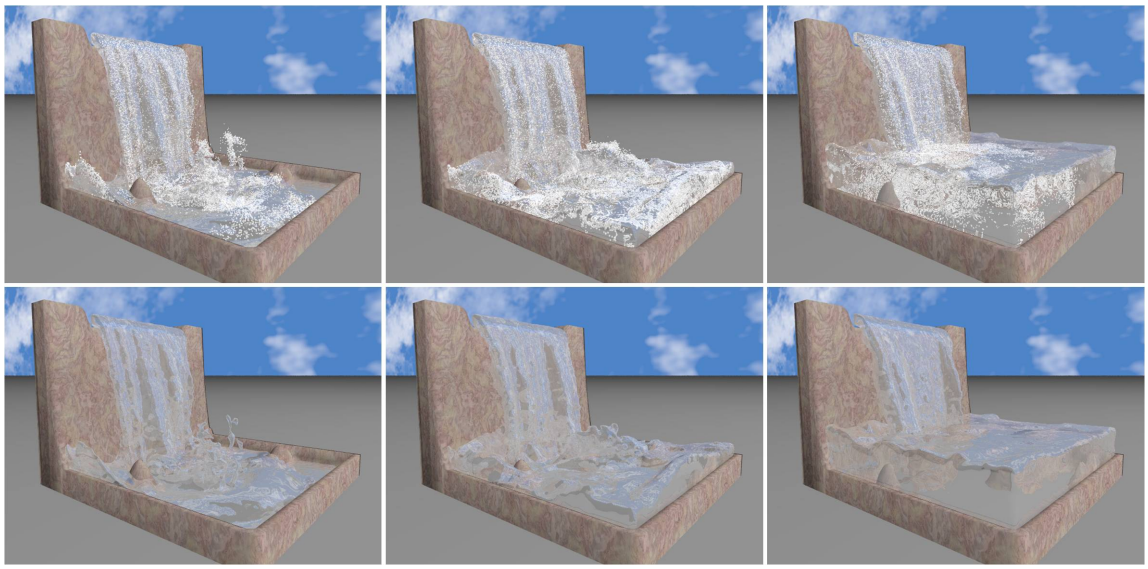


Figure 8: Waterfall. Liquid stream is pouring down into a tank of water, with generated spray, bubble, and foam (refer to the first row), and the second row shows the original simulation result of FLIP method.

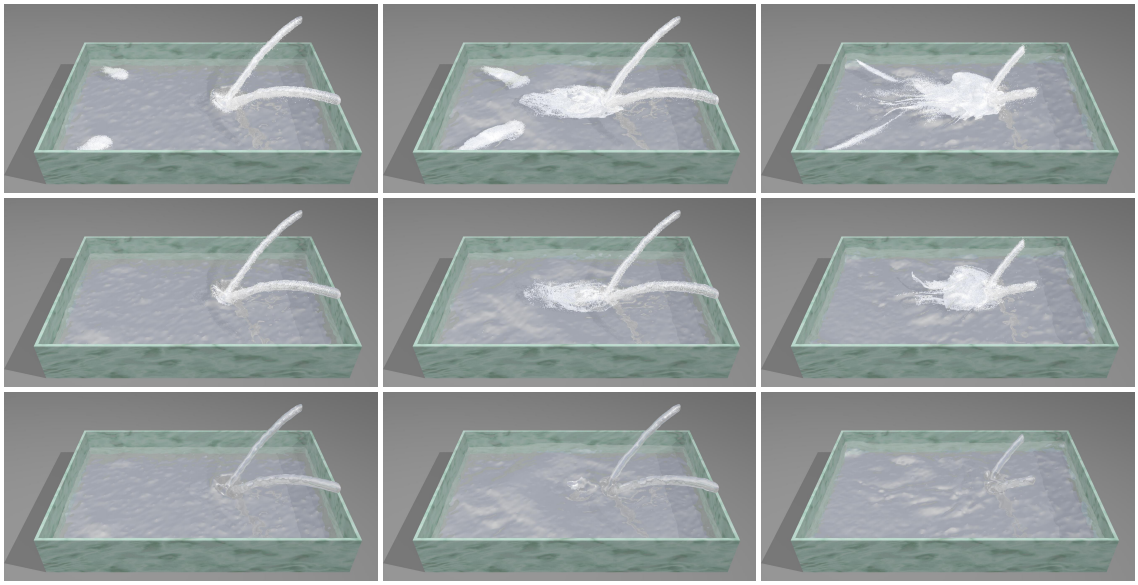


Figure 9: Pouring water into a water tank from the right (above the water level) and injecting water from the left (below the water level). Depth peeling is used in the first row, while the second row shows the result using the surface information only. The original FLIP simulation result is illustrated in the last row.

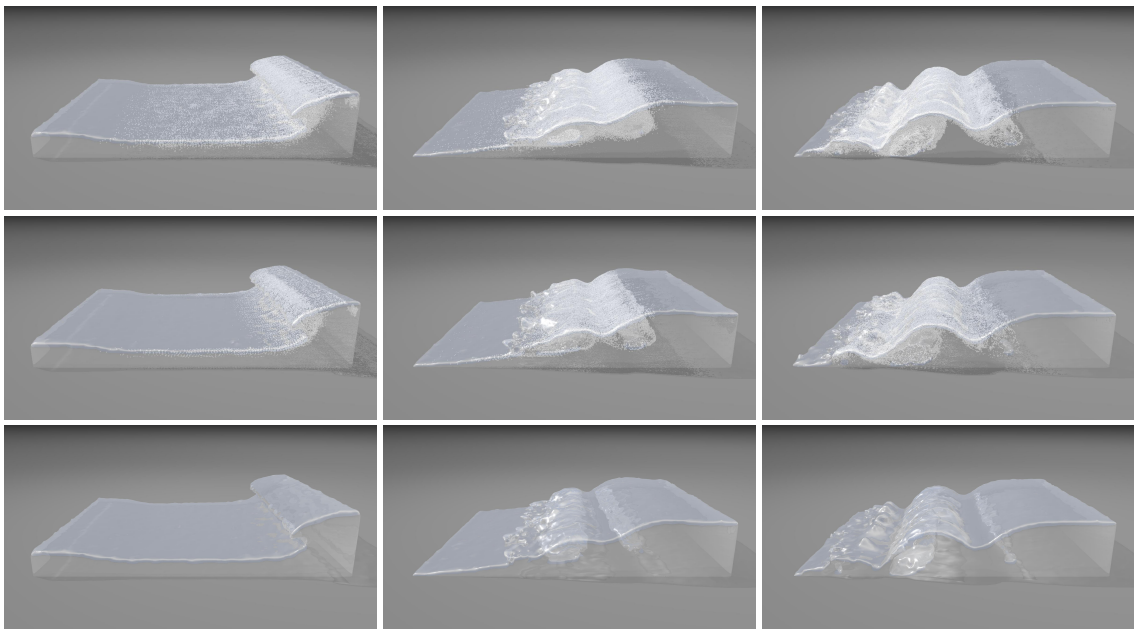


Figure 10: Sea waves are rolling over surfaces again and again in this scene. The first row illustrates the simulation results with depth peeling, but in the second row no layer has been peeled. In the third row, FLIP method is employed only.

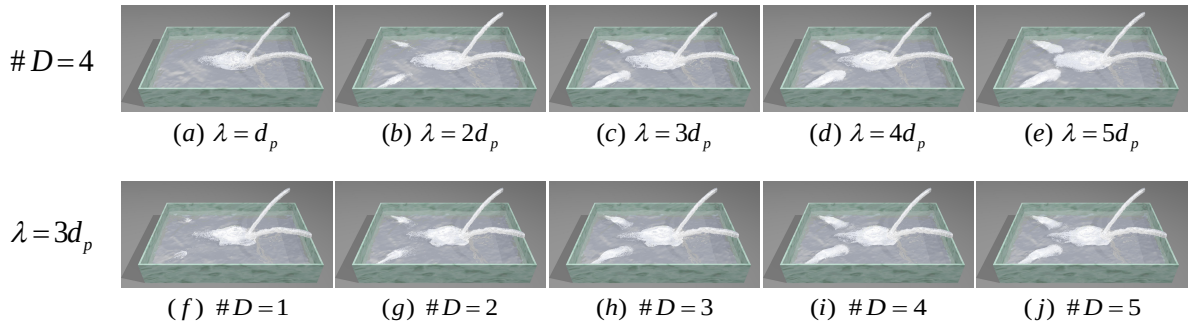


Figure 11: The first row shows the simulation results corresponding to different distance thresholds (λ) for one-time depth peeling. While the second row shows the simulation results corresponding to different passes of depth peeling ($\#D$) for one-time depth peeling.

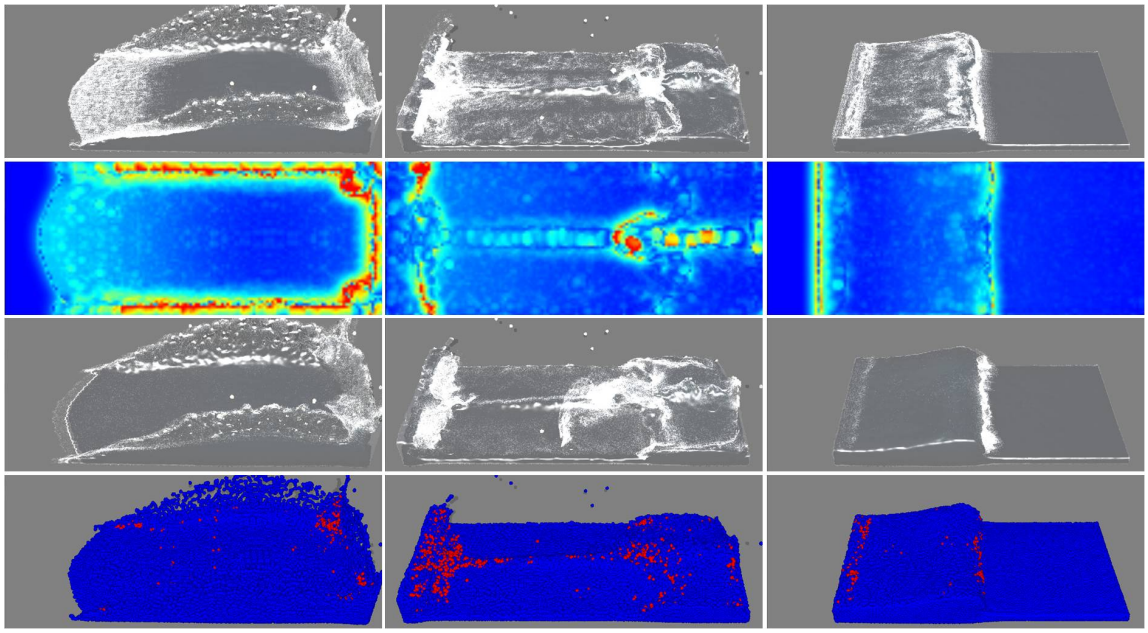
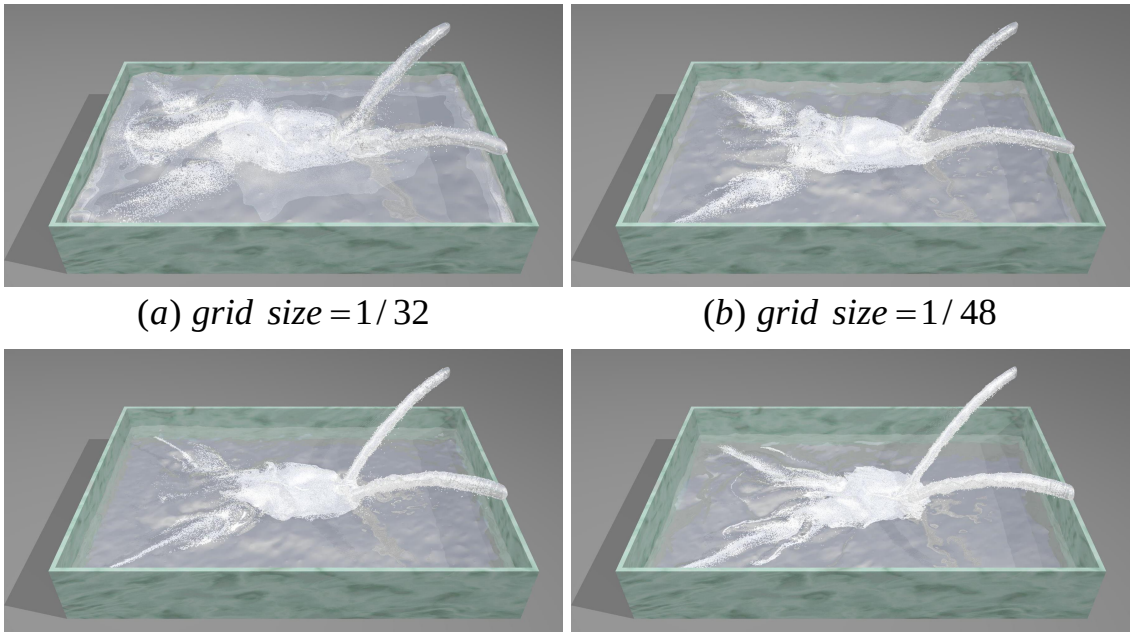


Figure 12: The comparison of the results from our paper (top two rows) and that from [2] (bottom two rows).



(a) $grid\ \ size = 1/32$ (b) $grid\ \ size = 1/48$

(c) $grid\ \ size = 1/64$ (c) $grid\ \ size = 1/80$

Figure 13: Comparison of the results based on different FLIP grid sizes. The total size of this scene is supposed as $1.5 \times 1 \times 1$. (a) The grid size is set as 1/32. (b) The grid size is set as 1/48. (c) The grid size is set as 1/64. (d) The grid size is set as 1/80.

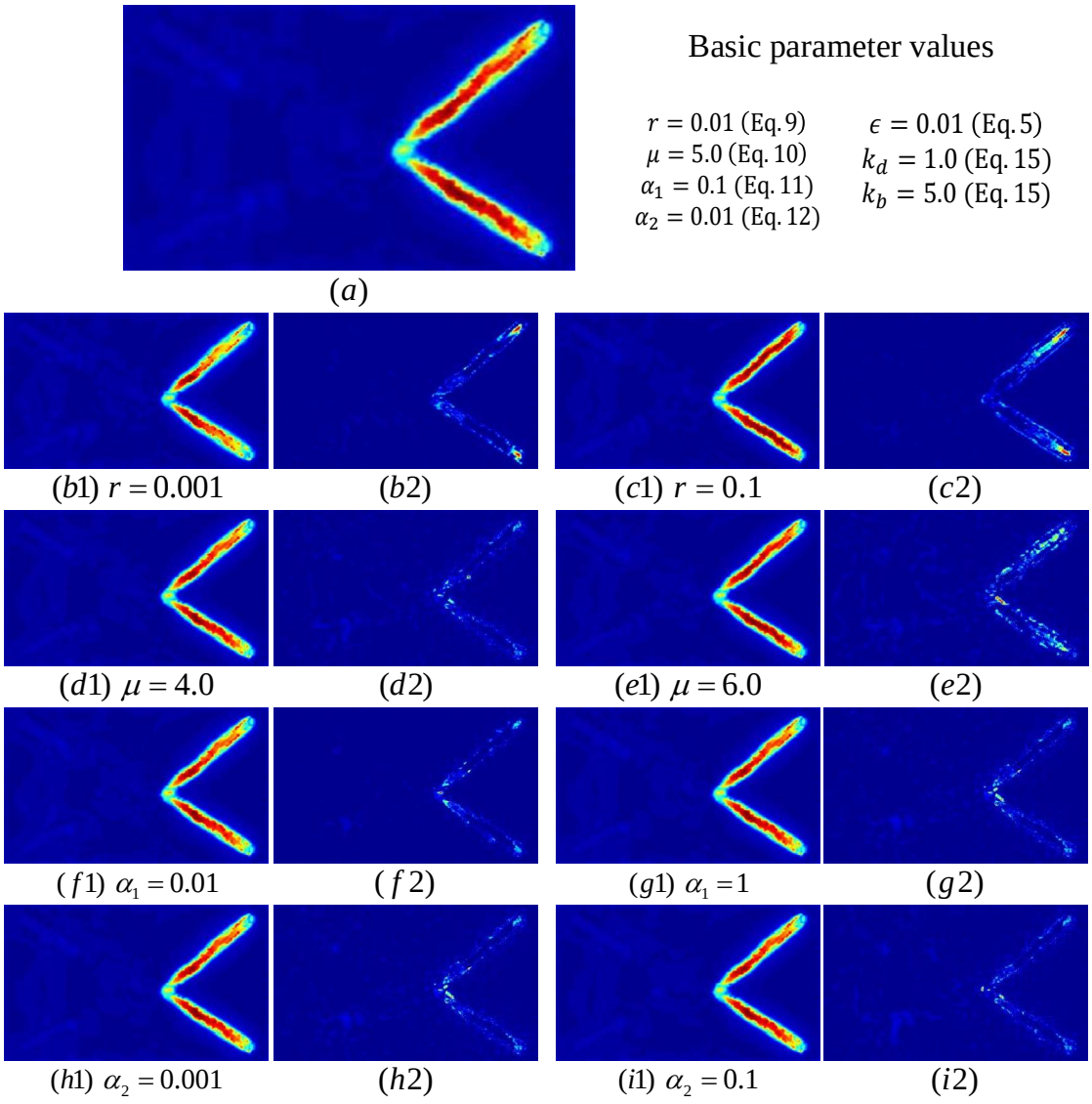


Figure 14: The logistic regression results of the first depth layer of frame 250 in pouring scene with different parameter settings. (a) The regression result with basic parameter settings, the parameter values are showed in the top right corner, the first and the third columns show the regression results with specific parameter settings (documented below the corresponding sub-figure), the second and the fourth columns show the difference between sub-figure (in the first and the third columns) and sub-figure(a). Warm color indicates high value, and vice versa. The basic and suggested parameter values used in the experiment are given in the top right corner.

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

Scene	#FP	GR	#DPM	DBR	FLIP	AN	DP	Other	#D	λ	AND	ANL	DPP
Dam-break (Fig. 4)	248.4k	96×48×64	288×144	108.2k	47.7	20.2	5.3	-	4	$3d_p$	82.8	19.2	9.6
Interaction (Fig. 7)	355.3k	96×48×64	288×144	49.6k	54.4	23.2	5.5	0.6	-	-	-	-	-
Pouring (Fig. 9)	242.1k	96×64×64	288×192	38.7k	41.5	18.4	5.3	-	4	$2d_p$	74.5	18.9	9.7
Waterfall (Fig. 8)	212.9k	64×64×64	192×192	40.8k	40.8	16.6	5.2	36.9	-	-	-	-	-

Table 2: Time performance(in milliseconds) with different parameters in depth peeling method in Pouring scene.

#D	λ	AN	DP	AND	ANL	DPP
1	$3d_p$	21.5	5.6	21.5	0	5.6
2	$3d_p$	20.1	5.8	41.3	17.6	6.2
3	$3d_p$	20.7	5.2	60.8	18.7	8.1
4	$3d_p$	20.2	5.3	82.8	19.2	9.6
5	$3d_p$	21.1	5.5	97.9	21.4	8.9
4	d_p	19.9	4.9	79.5	18.9	6.5
4	$2d_p$	20.4	5.1	81.8	19.8	7.6
4	$4d_p$	20.8	5.5	80.7	20.4	9.2