Contents lists available at ScienceDirect

Computers & Graphics

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

Particle Upsampling as a Flexible Post-Processing Approach to Increase Details in Animations of Splashing Liquids

ARTICLE INFO

Article history:
Received February 7, 2020

Keywords: Splash Modeling, Post-processing Approach, Procedural Method, Multi-scale, Multi-resolution, FLIP

ABSTRACT

Realistic and detailed liquid splashes require costly fine-scale discretization. We present an efficient post-processing approach for particle-based methods to locally improve the behavior of splashes on coarser liquids. Our method first computes a splash volume over time from the intersections between an identified upsampling volume and colliding volumes. We then upsample particles inside cells of the splash volume; these cells are pre-computed using a likelihood score based on criteria favoring emerging particles. In addition to the advection scheme, enhanced realism is achieved by applying a localized artificial pressure on upsampled particles in order to mimic surface tension in critical regions of splashes. Finally, we propagate waves using a novel implicit model that couples the impact of upsampled particles on the coarser liquid by updating the velocity field at these locations. Our implicit wave model can produce detailed swirls by solely applying velocity updates directly on the underlying particles from the coarse liquid, and prevents from using a high density of surface points. As a result, our approach can generate localized and parameterizable high-resolution splashes from solid-liquid and liquid-liquid interactions, and thus can simulate a wide range of unique and customizable splashes on top of an animated coarse liquid.

1. Introduction

In liquid simulation for visual effects, the discretization of the model is key to extract important features from complex examples, such as splashes and highly turbulent flows. This implies that in a particle-based liquid simulation, the number of particles must be strongly increased to bring out the fine details desired for the final outcome. With a uniform density of particles, most of the particles do not contribute to the desired and expected behavior displayed in the final rendered image, as they are either submerged below the surface or remain almost motionless throughout the simulation.

In this paper, we propose a flexible post-processing approach to improve splash modeling on top of a lower-resolution FLIP liquid surface. A subset of the input liquid volume is extracted from an input FLIP simulation. The extracted volume, so-called *splash volume*, is generated using the solid-liquid or liquid-liquid interactions occurring during the input animation. This volume is then augmented with a denser set of additional particles advected through the input velocity field. *Tracer* particles are seeded among the upsampled particles to control the behavior of the denser splash volume introduced by our approach. These tracers exert artificial forces, allowing the simulation of a surface-tension effect similar to applying an artificial pressure force. The collisions between airborne upsampled particles and the surface of the input coarse liquid are used as impact sources that inject energy into a surface-based linear wave model.

We present a novel and comprehensive splash modeling approach for particle-based liquids. Our main contributions can be summarized as the introduction of:

- A volume-based method to determine and upsample the splash zone for liquid-liquid and solid-liquid interactions.
- Localized tracers to induce diffuse and surface-tension behaviors on upsampled particles.
- A sparse implicit linear wave formulation to handle inter-

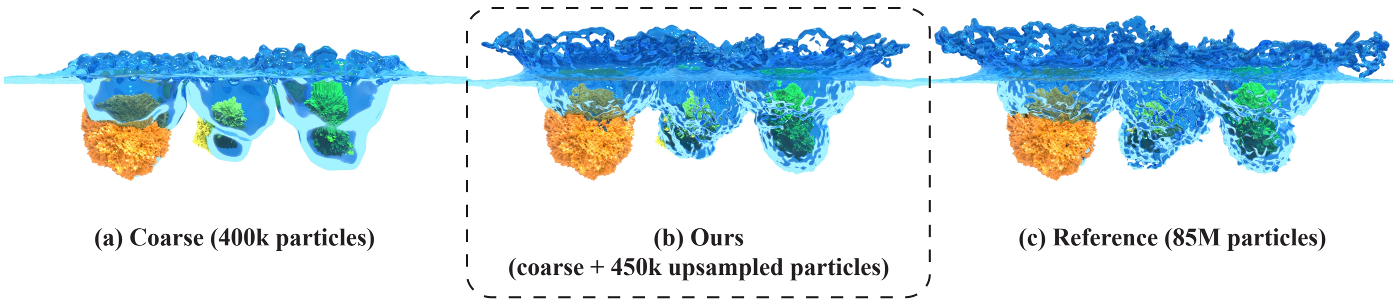
(a) Coarse (400k particles)
(b) Ours
(coarse + 450k upsampled particles)
(c) Reference (85M particles)

Fig. 1: Induced splash dynamics is generated (b) on top of a coarse FLIP liquid provided as input (a). Our approach is able to reproduce localized realistic splashes comparable to a high-resolution (about 100× more particles) reference (c) with solely 450k upsampled particles seeded within a small portion of the domain.

actions between the coarse input and the upsampled particles at impact locations.

- A trade-off between realistic and artistic splash effects.
- An interactive splash editing approach (once precomputation steps are performed).

2. Related Work

Over the years, several solutions to simulate realistic and detailed fluids have been proposed for purely Lagrangian methods, such as explicit and implicit SPH [1, 2, 3], and for hybrid methods, such as FLIP [4, 5, 6, 7, 8]. We refer readers to a comprehensive survey [9] and a book [10] describing in detail these methods. Procedural methods have also proven to be very efficient to enrich an input coarse fluid with fine details while offering a more practical artistic control. In the following, we focus primarily on topics more closely related to our work.

2.1. Upsampling Methods and Adaptive Models

Several approaches have explored different ways to increase the apparent spatial resolution of a coarse fluid simulation. In the last decade, the challenge of increasing the apparent resolution of fluid simulations has been addressed through many variants of sophisticated upsampling methods and spatial adaptive simulation models.

2.1.1. Upsampling Methods

These methods are able to effectively improve the high-frequency details while preserving the low-frequency behavior of the fluid, especially for smoke simulations [11, 12, 13, 14]. Although we focus solely on increasing the resolution of localized splash behaviors in liquids, our work shares this same goal. Few attempts to apply this goal to liquids were made for both Eulerian [12] and Lagrangian [15, 16] methods. Instead of upscaling the input velocity field, our method provides a realistic way to upsample localized portions of the unmodified coarse simulation data (particles and velocity field).

Some researchers have considered adding sub-scale quantities (e.g., curl noise) to existing coarse simulations [11, 12], while others have coupled these quantities to the Navier-Stokes equations [13]. In all these techniques, the quantities are extrapolated into the obstacles (geometric models) to prevent artifacts around them, but at the cost of significantly less realistic behaviors compared to high-resolution simulations. Selle et al. [17] proposed an approach to introduce particle-based vorticity to grid-based methods. Their work was extended by Pfaff et al. [14] to handle dynamic obstacles. In contrast to adding subscale quantities to existing simulations, other researchers have introduced multi-scale and adaptive formulations for particle-based methods that extract different ranges of details within a simulation domain [18, 19, 6, 20].

Kim et al. [21] introduced a novel Eulerian approach to increase the apparent spatial resolution of an existing liquid simulation. As they noted, increasing the free-surface details has shown to be a valid solution to the up-scaling problem on liquids, since the tangential velocity (and associated turbulence) is loosely coupled to the fluid velocity field. Their work was extended by Mercier et al. [22] to propose a coupled scheme using a high-resolution wave simulation while preserving the entire frequency spectrum of the input coarse simulation. For more details, we refer readers to the comprehensive survey by Thuerey et al. [23] on recent fluid up-scaling methods. These methods are unable to generate small-scale details and are limited to increase the actual apparent resolution. We offer a complementary and efficient way to recreate the missing small-scale details at sudden disturbance locations occurring with coarse liquids.

2.1.2. Spatial Adaptive Models

Even though our approach acts as a post-processing method, it is closely related to the essence of several spatial adaptive simulation models of the state of the art. Adjusting the spatial resolution of simulation models has been firstly introduced for fluid simulation by Desbrun and Cani [24]. Their approach adaptively handles highly deformable models through a refining step applied to the particle resolutions. Similarly, Adams et al. [25] proposed an adaptive sampling method focusing on the computational efforts of complex geometric regions of a fluid volume. They reduce the number of particles according to their visual contributions. Adaptive sampling on fluid simulations has also been used to improve and preserve the visual appearance of thin sheets of fluids. After identifying the thin portions of the fluid, the approach introduced by Ando et al. [6] resamples these critical regions based on the anisotropy of the particle neighborhoods. On a different note, Orthmann and Kolb [26] used a temporal blending technique to reduce the number of particles within low-resolution regions. Horvath and Solenthaler [27] improved the conservation of mass for resolution refining methods of particle-based fluid simulation. Meanwhile, Zhu et al. [28] proposed a new grid structure to extend the domain dimensions while preserving the fine details of a fluid simulation. More recently, Aanjaneya et al. [29] also introduced an efficient and sparse data structure to leverage the adaptivity of large-scale simulation domains. Recently, Sato et al. [30] extended the narrow band FLIP method to adaptively allow a transition between the particle band and the grid-based simulation anywhere in the domain (i.e., not exclusively close to the surface). Lastly, inspired by leveraging the data structure adaptively toward multi-scale fluid simulations, Ibayashi et al. [31] introduced a novel warping grid in order to dynamically deform a regular grid evolving during the simulation. While using spatial multi-scale resolution with particles improves visible surface details of liquids, their adaptive nature makes it difficult to exploit a trade-off between procedural and physics-based methods as our approach offers.

Although the resolution dependency is well known in the field of fluid simulation for computer graphics, some challenges remain. Re-computing the forces and the advection step of a small portion of the liquid within a given simulation could be significantly expensive and particularly demanding to preserve its stability. Some authors have shown that this can be improved by using a machine learning process [32, 33, 34, 35]. Compared to these data-driven methods, our approach only requires a coarse liquid simulation to generate high-resolution results. Closer to our motivation, procedural methods have proven that upscaling a simulation can produce interesting small-scale features [21, 22]. In contrast to these procedural methods, our approach is able to locally generate splash details that were poorly approximated due to a coarse discretization of the input simulation domain.

2.2. *Diffuse Material and Splash Modeling*

Several researchers have focused their work on handling interactions of air-liquid mixtures. The results of these interactions generate phenomena such as foam, spray, and white water at the interface of the liquid. Several papers have proposed hybrid schemes to handle sub-scale details with particles advected through a grid-based simulation [36, 37, 38, 39]. A purely Lagrangian approach was proposed by Ihmsen et al. [40]; they used post-processed layers of diffuse particles on top of an SPH simulation to represent foam and spray effects. In our method, we handle the diffuse portions of a liquid as ballistic particles and refine their behavior with localized artificial pressure corrections from seeded tracers among the upsampled particles.

The diffuse air-liquid interactions are more likely to happen within splashes and highly turbulent regions of liquids. Kim et al. [41] focused their efforts on identifying under-resolved regions and adding massless markers in splashing portions of the liquid. Chentanez and Müller [42] also proposed a solution to improve the visual appearance of liquids for real-time application. Their hybrid model uses spray particles to approximate the liquid-air interactions between rigid and soft bodies. Recently, Um et al. [34] introduced a novel data-driven method to increase the realism of diffuse effects on splashing liquids. A perceptual study was also published on these splashing behaviors [43]. However, in contrast with our work, we are focusing solely on improving the splash modeling before its diffuse effects. Our approach focuses on key events prior to splashing behaviors.

2.3. *Wave Simulation*

Many researchers have considered using a wave simulation instead of a computational fluid to simulate liquid motion [44, 45, 46, 47]. Through the years, they have presented several variations to reproduce waves with the capillary wave equation [44], the *iWave* model [46], and the shallow-water equation [48]. These methods were used to represent a very large body of liquid (e.g., an ocean) as well as to enrich fine-scale details in liquid simulations.

Particle-based waves were also used to increase high-frequency details of liquid simulations [47, 22, 49]. In these methods, the wave equation is used and seeded through advected particles. However, some of these approaches have experienced difficulties when handling complex scenarios such as splashing and highly turbulent behaviors [45, 47]. Mercier et al. [22] managed to robustly couple a dense point set wave representation to a smooth and meshless surface. Later, Yang et al. [49] proposed a novel implicit linear wave model to enrich SPH simulations. Recently, the dispersion kernels used with wave simulations were improved for real-time applications [50, 51]. Similarly, we handle interactions between the upsampled particles and the input coarse liquid with a point-based wave representation and update the input particle velocities and positions.

3. Method Overview

We briefly describe in this section each step of our pipeline (Fig. 2). As we aim to benefit from hybrid methods, we decided to rely on the widely known FLIP-based fluid that we use as a coarse input to our approach.

From a coarse simulation as input data (e.g., particles and a velocity field for each frame of the input simulation), two precomputation steps are performed. First off, we compute a splash volume (§ 4.1) by combining the intersection between user-defined colliding volumes and the input volume (i.e., generated from the coarse input liquid). As presented in Fig. 2, the splash volume (e.g., green portion of the upsampling step) is slightly extended (e.g., yellow portion of the upsampling step) to fully cover disturbances surrounding solid-liquid and liquid-liquid interactions. Thereafter, the resulting volume is discretized and upsampled using a higher density of particles per cell (§ 4.2). These upsampled particles are then injected into the existing velocity field and advected alongside the input particles of the coarse liquid (§ 5.3). Among upsampled particles, so-called tracers are seeded (§ 5.1) and used as control particles to apply localized artificial forces mimicking the effect of surface tension (§ 5.2). Lastly, we use a point-based linear wave model to couple interactions between the upsampled particles and coarse particles at impact locations. The waves are seeded to enrich

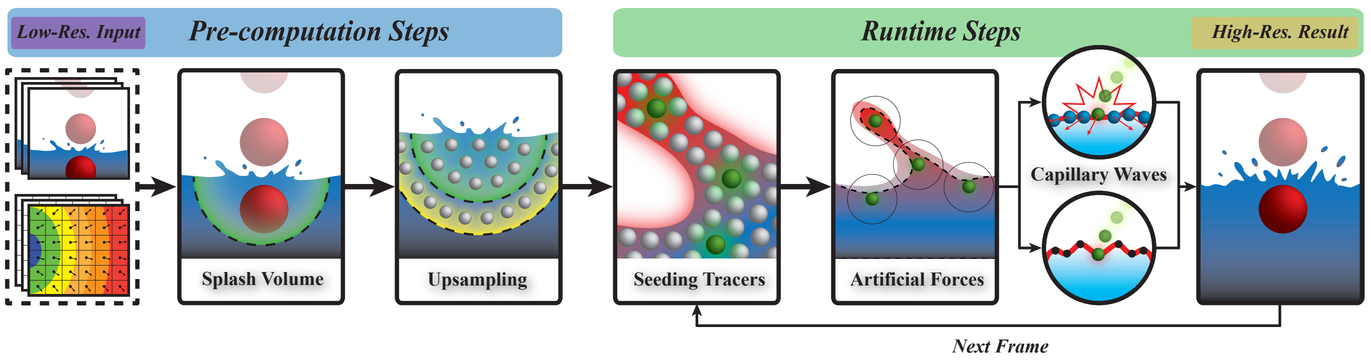

Fig. 2: Overview of our pipeline. The particles and the velocity field of an existing animated coarse liquid are used as initial inputs. The first two steps are part of a pre-computation process to determine and upsample a splash volume. The simulation steps iterate on the tasks to seed tracers among the upsampled particles, apply a localized artificial pressure, and generate linear waves at impact particle locations on the coarse liquid surface. The highly detailed splashes generated by our approach are computed at a much lower cost than would have been necessary to simulate similar splashes at high resolution with traditional approaches.

these impacts over the coarse input liquid (§ 6). As shown in Fig. 2, two pre-computation steps are performed only once for the whole input simulation data and can be reused with different parameter sets to generate alternative high-resolution results.

4. Splash Volume

A subset of the input simulation domain is determined to locally increase details around potentially splashing portions of the liquid. This volume is computed from the coarse input simulation data. At each frame of the input simulation, we identify a colliding volume that will contribute to the *splash volume* (§ 4.1). Thereafter, the splash volume is refined to upsample important portions of the liquid with more particles (§ 4.2).

4.1. Splash Volume Generation

From the input liquid, our method splits the scene into *upsampling* and *colliding* volumes. These distinct volumes are determined manually at the first frame of the input animation. The colliding volumes can be either a volume of liquid or a static or dynamic obstacle. We identify the upsampling volume as Ω^{up} and the colliding volumes as Ω^{c}.

The splash volume Ω^S is obtained from the unions of the upsampling volume Ω^{up} and the colliding volumes Ω^c over time. The splash volume Ω^S is therefore pre-computed from all N_f frames of the input simulation and is expressed as

$$\Omega^S = \bigcup_{i=1}^{N_f} \left(\Omega_i^c \cap \Omega_1^{up}\right), \tag{1}$$

which corresponds to the intersection between the upsampling volume Ω^{up} and the union of the colliding volumes. The updating process of the splash volume Ω^S is illustrated in Fig. 3 as well as in the accompanying video. As shown in the top row, we only keep the intersection with the volume Ω^{up} at the first frame $i = 1$.

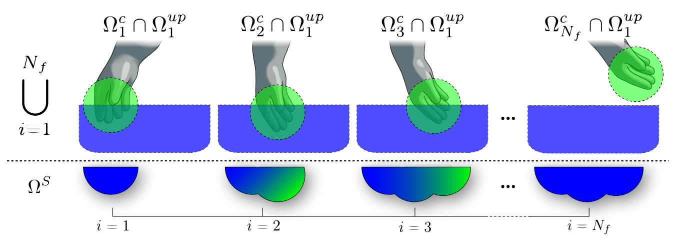
$\Omega_1^c \cap \Omega_1^{up}$
$\Omega_2^c \cap \Omega_1^{up}$
$\Omega_3^c \cap \Omega_1^{up}$
$\Omega_{N_f}^c \cap \Omega_1^{up}$
$\bigcup_{i=1}^{N_f}$
...
Ω^S
$i = 1$
$i = 2$
$i = 3$
$i = N_f$

Fig. 3: Generation of the splash volume from the upsampling volume (blue) and the collision volumes (green).

Our method for generating a splash volume handles scenarios in which solid-liquid or liquid-liquid collisions occur. For solid-liquid collisions, we convert the input mesh of the static or dynamic obstacle into a volume Ω^c. As shown in Fig. 3, only the intersection with Ω^{up} is kept since it will ultimately represent the splash volume Ω^S to be upsampled. For liquid-liquid interactions, we split the volume of the input liquid into distinct volumes: one volume Ω^{up} to be upsampled (e.g., liquid container), and volumes Ω^c that will collide with the upsampled volume (e.g., a falling drop of liquid). For this type of interaction, we keep track of the particles initially composing Ω^c to generate the deforming volume Ω_i^c over time.

4.2. Particle Upsampling

The purpose of the oversampling step is to identify regions within the splash volume Ω^S in which upsampled particles should be added (see § 7 for implementation details). As opposed to upsampling entirely the volume Ω^S, our upsampling method focuses on regions (i.e., subset of cells) in which significant changes are likely to occur (see § 8.6 for a comparative analysis). Our *emerge-tendency* upsampling method concentrates on the regions of the splash volume in which particles are most likely to emerge. We define the emerge-tendency of cells as their probability to contain particles that reach the surface at some point during the entire animation. A likelihood score is computed for each cell identifying which ones to upsample. The likelihood score is obtained by computing three reward terms as follows:

$$\Theta(x) = \sum_{i=1}^{N_f} \mathrm{rt}_i(x) + \alpha \sum_{i=1}^{N_f} \mathrm{ru}_i(x) + (1-\alpha) \sum_{i=3}^{N_f} \mathrm{rs}_i(x), \qquad (2)$$

where Θ is a set of cells containing the score associated with each grid cell x within Ω^S and in which the reward terms are respectively based on particle trails rt, velocity norm $||\vec{u}_i(x)||^2$ noted as ru, and velocity orientation stability rs. We also introduce a weighting factor α to the reward terms ru and rs controlling the relative influence of each other on the likelihood score. We used $\alpha = 0.5$ in all our examples except with the *River* (see § 8.2). Given as input the entire animation of

Algorithm 1: Identify cells to upsample with Ω^S.

```
for $x \in \Omega^S$ do
    $\mathrm{rt}(x) = \mathrm{ru}(x) = \mathrm{rs}(x) = 0$
    for $i \in [1, N_f]$ do
        if crossSurface($p(x)$, $i$) then
            $\mathrm{rt}(x) \mathrel{+}= \mathrm{rt}_i(x)$
        $ru(x) \mathrel{+}= ||\vec{u}_i(x)||^2$
    for $i \in [3, N_f]$ do
        $\mathrm{rs}_i(x) = 1$ for $j \in [i-2, i]$ do
            $\mathrm{rs}_i(x) \mathrel{*}= \max(\vec{u}_i^j(x) \cdot \vec{n}_i(x), 0) \geq \cos\frac{\psi}{2}$
        $\mathrm{rs}(x) \mathrel{+}= \mathrm{rs}_i(x)$
    $\Theta(x) = \mathrm{rt}(x) + \alpha\mathrm{ru}(x) + (1-\alpha)\mathrm{rs}(x)$
meanScore = computeMeanScore($\Theta$)
$\mathbf{s} = \emptyset$
for $x \in \Omega^S$ do
    if $\Theta(x) \geq$ meanScore then
        $\mathbf{s} \leftarrow \mathbf{s} \cup x$
UpsampleCells($\mathbf{s}$, $s_{up}$)
```

the coarse liquid, computing the particle trail reward term rt is fairly trivial. We set $\mathrm{rt}_i(x)$ to be equal to the number of particles $p(x)$ originating from cell x that cross a surface cell within the splash volume at frame i. The surface cells are defined as the discretization of the surface level set. Since our objective is to predict which particles among the upsampled particles will reach the surface, only computing the trails of the coarse particles will be insufficient. In that sense, the input velocity field $\vec{u}$ complements the particle trails. We set $\mathrm{ru}_i(x)$ to the L^2 norm of the grid cell velocity. Consequently, the velocity L^2 norm term penalizes cells containing a lower velocity over time. Lastly, we complement the likelihood score Θ of each cell x with the reward rs, which considers the grid cell velocity orientation over time. The rs term rewards grid cells with velocity orientations pointing toward the liquid interface for a sequence of consecutive frames. The velocity orientation stability $\mathrm{rs}_i(x)$ is equal to 1 if consecutive velocities lie within a cone defined by an angle ψ aligned with the normal of the liquid interface (otherwise $\mathrm{rs}_i(x) = 0$). We figured out (after a few experiments) that a window of three consecutive frames (i.e., $i-2$, $i-1$, and i) was a good compromise to estimate our stability term over time.

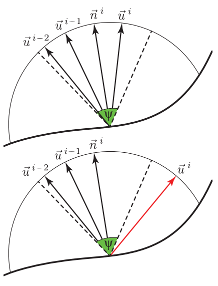

Fig. 4: 2D example of rs_i evaluation.

Consider the 2D example shown in the inset Fig. 4: the current and two previous velocities are oriented inside the cone defined by ψ in green (top image: $\mathrm{rs}_i(x) = 1$). The bottom image shows a case where one of the velocities $\vec{u}^{\,t}$ (red) is outside the cone (i.e., $\mathrm{rs}_i(x) = 0$).

The resulting subset of cells $\mathbf{s}$ to upsample contains the cells with a likelihood score (as shown in Alg. 1) greater or equal to the average likelihood score among the cells contained inside the splash volume Ω^S. Finally, the cells identified are upsampled using the commonly known jittered grid technique. As proposed by Zhu and Bridson [4], we create randomly jittered particles in every cell using d^3 subgrid positions where d is the subgrid dimensions. The number of particles per cell used in our examples corresponds to the number of particles per cell given from the coarse input liquid multiplied by a factor s_{up}. Therefore, the number of particles upsampled is equal to $s_{\mathrm{up}}d^3$.

5. Tracers and Upsampled Particles

Tracers are seeded in the splash volume, but only within the narrow band (§ 5.1) defined from the interface of the liquid. These tracers are used to influence the upsampled portions of the liquid; they mimic the effect of surface-tension forces observed in splashes. We call them *tracers* because they trace trajectories for upsampled particles. The movement of a tracer is initialized with the velocity transferred from the closest upsampled particle, and afterward, it is only influenced by gravity. The upsampled particles within a user-defined maximum distance of tracers are affected by an artificial pressure term as used with position-based fluids (PBF) [52]. The details of upsampled particles velocity update are presented in § 5.3.

5.1. Seeding Tracers

Since our aim is to enhance surface details around localized portions of the input liquid, we constrain the seeding of tracers within a narrow band according to a set of criteria. The seeding operation is performed in a way to achieve a uniform distribution, to distribute the tracers close to the interface, and to seed them according to the velocity norm throughout the animation. Tracers are seeded using the Fast Poisson Disk sampling method [53] within a narrow band of the splashing liquid. The narrow band is defined as a set of cells discretized from the liquid interface. We use the 3D implementation of the sampling method considering the narrow band thickness (see § 7 for implementation details). The band thickness $\Delta\tau_{\mathrm{tp}}$ is defined as twice the influence radius of the tracers (refer to Table 1). This band thickness was chosen to ensure a minimum number of upsampled particles affected by our tracers and to avoid neighborhood deficiency.

The last criterion on the velocity norm is used to discard insignificant motionless portions of liquid. The velocity norm criterion focuses on seeding tracers in regions where the input liquid is likely to be disturbed and to present notable small-scale detail differences. Tracers are seeded at locations where the L^2 norm of the velocity is increasing. In other words, tracers are seeded in cells with increasing velocities over time. The moment of creation of tracers is determined by the velocity criterion as well. To trigger the seeding process, we use a five-frame temporal sliding window $\triangle w_{\mathrm{tp}}$ and evaluate the velocity L^2 norm change for every grid cell. The L^2 norm evaluation of the velocities is performed as shown in Alg. 2.

Algorithm 2: Evaluate if the velocity L^2 norm of a cell at time t satisfies the tracer seeding conditions within the temporal frame interval $[t, t + \triangle w_{\mathrm{tp}}[$.

```
for $i \in [t, t + \triangle w_{tp}[$ do
    if $\|\vec{u}_{i+1}\|^2 \leq \|\vec{u}_i\|^2$ or $\|\vec{u}_i\|^2 < \overline{\|\vec{u}_i\|}^2$ then
        return false
return true
```

Although evaluating velocities within a five-frame window leads to very good results on our examples, this frame interval could be adjusted by the user. Cells where tracers will be seeded must also have a high velocity; the velocity L^2 norm must be higher or equal to the current average velocity $\overline{\|\vec{u}_i\|}$ in order to exclude small velocities increasing over time. We update an average velocity from all the cells containing particles at each frame and use it as a reference. Seeding tracers is computed at a negligible cost since it is performed exclusively on cells at the interface and inside the splash volume.

5.2. Localized Artificial Forces

We use the tracers to apply a correction term on the position of the upsampled particles to avoid artifacts such as clustering and clumping. These artifacts issued from negative pressures occur at the initialization step with the input velocity field. We use an approach similar to that of Monaghan [54] to correct tensile instability with an artificial pressure term, as follows:

$$s_p = -k \left(\frac{W(\vec{p}_i - \vec{p}_j, h)}{W(\vec{p}_i - \vec{q}, h)} \right)^n, \tag{3}$$

where $\vec{q}$ is a point inside the smoothing domain defined between $\vec{p}_i$ and $\vec{p}_j$, n the pressure power constant, and k a stiffness constant (we always use $k = 0.1$ and $n = 4$). The particle displacement $\triangle\vec{p}_i$ based on the artificial pressure term s_p is then expressed as follows in the particle position update step, as used by Macklin and Müller [52]:

$$\triangle\vec{p}_i = \frac{1}{\rho_0} \sum_j s_p \nabla W(\vec{p}_i - \vec{p}_j, h). \tag{4}$$

The support radius h_{tp} is defined between 0 and the input support radius h to adjust the influence of tracers over the upsampled particles. The value of h_{tp} depends on the desired visual outcome (as shown in Fig. 16). Furthermore, the particle density is usually unknown when using the FLIP simulation method. We thus derived a normalized expression implicitly taking density into account [55]:

$$\triangle\vec{p}_i = \frac{1}{\sum_j W(\vec{p}_i - \vec{p}_j, h)} \sum_j s_p \nabla W^*_{\mathrm{tracer}}(\vec{r}_{ij}, h_{\mathrm{tp}}). \tag{5}$$

Similarly to Müller et al. [56], we use the *Poly6* kernel for density estimation and the *Spiky* kernel for gradient estimation. In Eq. 5, we employ a slightly different kernel function to compute the position update. We propose a tracer-relative kernel function as follows:

$$W^*_{\mathrm{tracer}}(\vec{r}_{ij}, \vec{r}_{it}, h_{\mathrm{tp}}) = \begin{cases} \left(h_{\mathrm{tp}} - \min(|\vec{r}_{ij}|, |\vec{r}_{it}|)\right)^3 & 0 \leq |\vec{r}_{it}| \leq h_{\mathrm{tp}} \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $|\vec{r}_{ij}|$ is the length of the vector between upsampled particles i and j, and $|\vec{r}_{it}|$ is the length of the vector between upsampled particle i and the closest tracer. By promoting the closest tracer, we ensure to improve a localized particle attraction based on the most significant contributor. In the proposed approach, tracers are used as control particles over the upsampled particles. Their kernel-based contribution influences the neighboring upsampled particle positions by updating their velocities and by applying an artificial force to mimic the pressure nature of liquids on diffuse particles.

By comparison to position-based fluids (PBF) (Eq. 3), we use a point $\vec{q}$ defined as the position of the closest tracer. In contrast with the one used by Macklin and Müller [52], our artificial pressure term is primarily applied to particles within a radius distance from the seeded tracers. The distance between an upsampled particle and the closest tracer must be smaller or equal than the support radius h used for the coarse input. This term improves the generation of fine surface-tension details among the upsampled particles.

Our tracer-relative kernel function can also be easily integrated into the PBF model. As shown in Fig. 6, seeding tracers among the particles and applying our constrained artificial pressure can locally improve affected regions in a PBF liquid.

5.3. Upsampled Particles: Advection and Lifespan

During their lifespan, upsampled particles evolve between different states, based on the types of details they are likely to add to the simulation. In this section, we will explain the three states (*bulk*, *ballistic*, and *impact*) and how particles evolve between states (Fig. 5).

The bulk state is flagged when an upsampled particle is inside the splash volume. By default, all upsampled particles are considered as bulk at the initialization step (i.e., when seeded at precomputation steps). The bulk particles p_{bulk} are advected using a Runge-Kutta scheme through the input velocity field [4]. The velocity update used for bulk particles is performed solely using the input coarse velocity field as follows:

$$\vec{u}^{\,i}_{p_{bulk}} = \vec{u}^{\,i-1}_{p_{bulk}} + \mathrm{lerp}(\triangle\vec{u}^{\,i}_{input}, \vec{x}^{\,i}_{p_{bulk}}). \tag{7}$$

As with FLIP, the weighted average of the velocities of the upsampled particles is computed and added to the nearby staggered MAC grid $\vec{u}^{\,i}_{input}$ nodes. At each step, the interpolated difference from the input grid velocity $\triangle\vec{u}^{\,i}_{input}$ is computed and assigned to each particle at their respective position $\vec{x}^{\,i}_{p_{bulk}}$. This velocity update step is very effective since we are computing these velocity changes only for cells inside a subset (i.e., splash volume) of the input simulation domain.

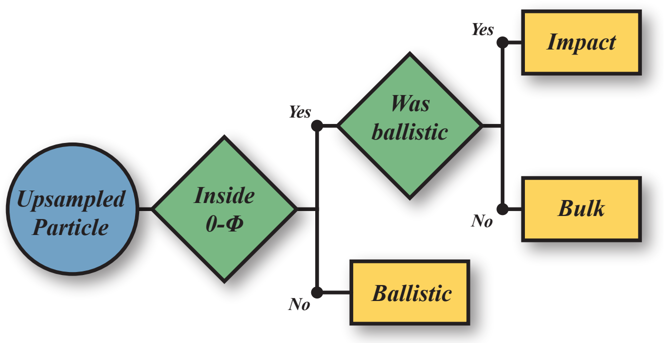

Fig. 5: State machine (trigger events in green) for determining the particle state (yellow).

The ballistic state is triggered when an upsampled particle leaves the bulk volume determined by the liquid interface of the input simulation (see Fig. 5). The upsampled particles flagged as ballistic are initialized with the current particle velocity $\vec{u}^{\,i}_{p_{bulk}}$ previously computed before leaving the coarse liquid (i.e., w.r.t. the liquid surface). They are later influenced by gravity and possibly by tracers (as previously described in § 5.2) as follows:

$$\vec{u}^{\,i}_{p_{bal}} = \vec{u}^{\,i-1}_{p_{bal}} + \vec{g}\triangle t \qquad (8)$$

and where the updated position of a bulk particle $\vec{x}^{\,i}_{p_{bal}}$ at frame i is obtained by integrating the resulting $\vec{u}^{\,i}_{p_{bal}}$ with the correction term defined in Eq. 5.

Lastly, the impact state is triggered when a ballistic particle intersects back with the coarse liquid surface. The name *impact* is taken from the fact that these particles will alter the velocity of the particles within the bulk volume. This state is only flagged for a particle during one frame. It allows us to detect the positions for seeding point-based waves within the bulk portion of the input liquid (§ 6). As soon as we store the impact position, the particle is flagged back as bulk and its velocity is updated according to the underlying velocity field. We use tri-linear interpolation for a smooth transition of the velocity from its ballistic nature to the current grid cell.

For efficiency reasons, we discard particles that are unlikely to contribute to splashes. We define an upsampled particle as discarded if it no longer contributes to the surface details of the liquid. We use two criteria to determine if an upsampled particle contributes to surface details: its distance to the interface inside the liquid and its current velocity. With these two criteria, we manage to preserve small- and large-scale details around induced splashes. The distance criterion uses the same distance as defined when seeding tracer particles (§ 5.1). However, this threshold does not apply if the upsampled particle is still inside the splash volume. As a second criterion, we also discard upsampled particles if their velocity is lower than the average velocity. We use the same average velocity per cell obtained when seeding tracers (see § 5.1).

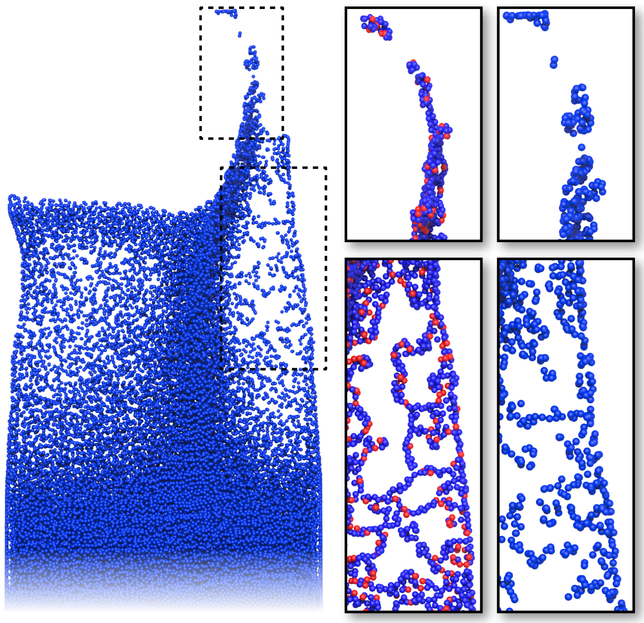

Fig. 6: Our tracers (red) constrain an artificial pressure term integrated with PBF. The localized correction produces a smooth and controllable surface-tension effect among the upsampled particles (left) compared to the originally diffuse behavior (right).

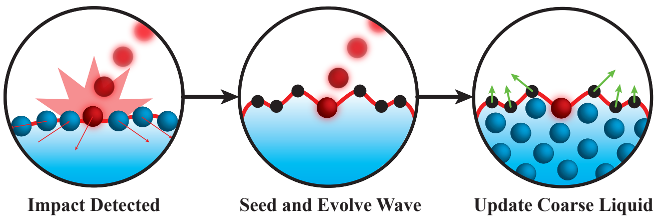

Fig. 7: An impact is defined by the intersection between an impact particle and the coarse liquid (left). Surface points distributed and projected on the surface are used to support and evolve the linear waves (middle). The tangential velocities of the point-based waves are used to update the underlying grid velocity, and then the positions of the input particles from the coarse liquid (right).

6. Implicit Linear Waves

Since ballistic particles are likely to touch the interface of the coarse liquid, additional details must be generated on the liquid interface in order to preserve the realism of the approach. Following the detection of impact particles (i.e., ballistic particles intersecting the coarse liquid surface), we handle their interactions with the coarse liquid provided as input (Fig. 7). In our approach, these interactions are reflected as velocity updates on the grid cells, and generated as seeded waves evolving from the impact locations. These waves are modeled by surface points uniformly distributed to cover the implicit surface. A wave displacement is computed according to the impact properties and the estimated normals obtained from the implicit surface. The evolving waves locally affect the neighboring cell velocities by their tangential velocities. The neighboring affected area is determined by the impact radius. The positions of the coarse input particles are then updated to reflect these changes (see Fig. 8).

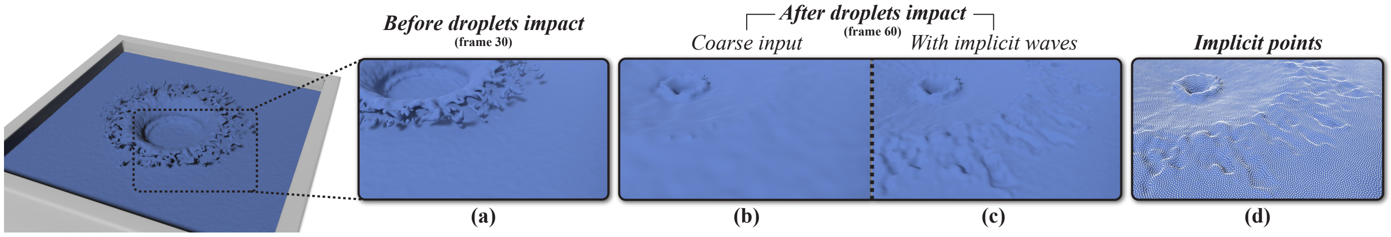

Fig. 8: The small droplets of a splash (a) are often absorbed on impact (b), resulting in loss of multiple fine surface details. Our implicit linear wave model provides convincing interactions through small swirls generated by these droplets (c). The waves are propagated through the implicit points projected on the surface (d).

6.1. Surface Points

The surface points are distributed uniformly over the entire implicit surface generated from the coarse liquid. These surface points are generated once at the very first frame of the input animated fluid. We use a Poisson distribution to initialize the surface points along the surface. The positions of the surface particles from the coarse input liquid are used as the initial locations for Poisson disk sampling [53]. The surface points that we call *implicit points* from this stage, are then projected on the surface of the coarse liquid. The number of implicit points created is bounded by the particle density per cell of the input liquid. Since we aim to solely introduce small impact interactions, our wave model does not require a computationally expensive maintenance process (as opposed to Mercier et al. [22]) on implicit points after the projection step. In addition, it is noted that surface points are not used when generating the surface mesh (as shown in Fig. 7(c)) considering that their sole purpose is to perturb surrounding cells of the input velocity grid.

6.2. Wave Seeding and Propagation

The seeding locations are determined by the position of the upsampled particles flagged as impact particles. We use the current velocity magnitudes of the upsampled particles at impact locations to parameterize the wave properties when seeded. Since our goal is to infer interactions from these impacts, our wave model is defined as an efficient gridless version of the 2D heightfield liquid simulation [57]. With this model, height is the factor that determines the wave velocity. On impact, we update the heights $h_{p_{\text{impl}}}$ of the implicit points (p_{impl}) i within radius as follows:

$$h^i_{p_{\text{impl}}} = h^i_{p_{\text{impl}}} + \frac{1}{N}\sum_{j=0}^{N-1}\left[\cos\left(\frac{\|\vec{c}-\vec{p}^j\|}{r_{\text{im}}}\pi\right)\alpha_s\|\vec{u}_{\text{im}}\|\right], \tag{9}$$

where $\vec{c}$ is the center of the impact position, $\vec{p}^j$ the position of the j^{th} implicit neighbor of N points, and $\vec{u}_{\text{im}}$ the velocity of the impact weighted by a user-defined normalized scaling factor α_s. The radius of the impact r_{im} is also defined as a function of the impact velocity norm $\|\vec{u}_{\text{im}}\|$ as follows

$$r_{\text{im}} = \alpha_r\|\vec{u}_{\text{im}}\|, \tag{10}$$

and weighted by a user-defined normalized constant α_r. In our examples, we used an α_r equal to 10% of the simulation scale from the coarse input liquid. The first term in brackets of Eq. 9 is the height displacement applied by an implicit neighbor point j, and the second term is basically the strength of the wave displacement.

We propagate and evolve the waves using the neighbor contributions at each frame. For each implicit point i, an average height $\bar{h}_{p_{\text{impl}}}$ from the neighborhood is computed and used to update a local 1D wave velocity (initialized as zero before impact):

$$u^i_{p_{\text{impl}}} = u^i_{p_{\text{impl}}} + \frac{1}{N}\sum_{j=0}^{N-1}\Big[(1-\beta_{\text{damp}})u^j_{p_{\text{impl}}} + 2(\bar{h}_{p_{\text{impl}}} - h^j_{p_{\text{impl}}})\triangle t\Big]. \tag{11}$$

The updated height is obtained by adding this 1D velocity to its current value $u^i_{p_{\text{impl}}}$ (initialized by the grid velocity at that position). The term $(1-\beta_{\text{damp}})$ enforces wave damping. Finally, we update the positions on every implicit point affected, and according to the orientation of the implicit point normal . The normal at these points is computed using an averaged least-squares planar fit of the local gradient of Φ based on the implicit point neighbors. The planar fit allows us to compute a plane defined by a tangent and a bi-tangent, both orthonormal to its normal.

6.3. Influence over Coarse Particles

We achieve the update on the input surface particle positions by converting the affected particles to bulk particles. The subset of affected particles is determined by the impact radius. As a wave is affected by damping, the corresponding particles return progressively to their current input state. The implicit point velocities $u_{p_{\text{impl}}}$ are used to alter the coarse grid velocity, and to update the affected coarse particles. We use the tangential velocity at each implicit point defining the waves. The tangential velocity is computed by orienting the implicit point velocity towards its tangent vector. For our wave model, we chose the tangent vector $\vec{t}^{\,i}_{p_{\text{impl}}}$ according to the impact velocity orientation and with respect to $\vec{n}^{\,i}_{p_{\text{impl}}}$. We compute the updated grid cell velocity by interpolating it between its current value and the tangential velocity at that position (i.e., $\vec{x}_{p^i_{\text{impl}}}$). The grid cell velocity is returning to its unmodified value as the tangential velocity gets attenuated as:

$$\vec{u}^{\,*}_{\text{input}}(x) = (1-\gamma_{\text{att}})\vec{u}_{\text{input}}(x) + \gamma_{\text{att}}(u^i_{p_{\text{impl}}}\vec{t}^{\,i}_{p_{\text{impl}}}). \tag{12}$$

The normalized interpolation term γ_{att} is calculated based on the ratio between the current implicit point velocity at frame t

and the one when the wave has been seeded ($\vec{u}^0_{p_{\text{impl}}}$):

$$\gamma_{\text{att}} = \frac{||\vec{u}_{p_{\text{impl}}}||}{||\vec{u}^0_{p_{\text{impl}}}||}. \tag{13}$$

The velocity updates are performed exclusively on the surface cells (i.e., according to the discretized level set).

7. Implementation

Most of the steps of our approach are computed on the GPU with CUDA. Our examples were generated on an Intel i7 quad core running at 3.4 GHz with 16 GB of RAM, and using an NVIDIA GTX 1080 with 8 GB GDDR5X. Our input coarse simulations come from the implementation of FLIP available in Houdini 16.5 [58]. Although our input is obtained from a traditional FLIP method, the use of any of its variants, such as narrow-band FLIP [7], would have been equally valid. The parameter values used with the presented examples are exposed in Table 1. In the following, we also include implementation details for core steps of our method to facilitate reproducibility of the presented results.

Parameter	Notation	Value
Input support radius	h	$*$
Scaling reward terms	α	$[0.5, 0.7]$
Subgrid dimensions	d	$\boxed{2, s_{up}d]}$
Tracer influence radius	h_{tp}	$[0, h]$
Tracer band thickness	$\triangle\tau_{\text{tp}}$	$2h$
Tracer temporal window	$\triangle w_{\text{tp}}$	5
Tracer pressure power	n	4
Tracer stiffness constant	k	0.1
Upsample factor	s_{up}	$[2, 6]$
Scaling impact velocity	α_s	0.5
Scaling impact radius	α_{r}	$0.1 s_{\text{up}}$
Scaling wave damping	β_{damp}	$[0.7, 1.0[$

Table 1: Parameters used to generate our results. As exposed, our approach has only few user-defined parameters. Most of the parameters presented are fixed (i.e., invariant from the types of scenario) or given from the input liquid. * The input support radius h and scaling upsample factor s_{up} are the only parameters that are specific to a scenario; they depend on the scale and the level of detail required by it.

Splash Volume. When generating the splash volume, Ω^{up} and Ω^c are computed using an SDF of the input particles. During SDF computation, we also consider as boundaries the input obstacles such as static and dynamic meshes. The discretized splash volume resolution corresponds to the resolution of the input velocity grid. It is formed as a subset of cells of the input velocity grid. Of course, we solely consider cells containing particles as part of this subset. In the case of liquid-liquid interactions, we keep track of the distinct volumes of liquid by assigning the particle unique IDs to a volume ID. Naturally, and for that reason, we avoid using a resampling step when generating the input liquid simulation. For both solid-liquid and liquid-liquid interactions, we expand the colliding volume Ω^c in order to capture properly the disturbance on the upsampling volume Ω^{up}. Expanding Ω^c is particularly helpful with solid-liquid interactions since obstacles tend to push particles outside their boundaries. Moreover, we can automatically trigger to stop updating the splash volume Ω^S if it remains constant after several consecutive frames (e.g., as with the *Hulk's wrath* example shown in Fig. 13). We observed that while we should limit the number of cells in the splash volume, the impact on the total computations is less critical than other parts of our method. Lastly, we determine the first frame i from which we track the splash volume as the first collision detected between the colliding volume Ω^c and the upsampling volume Ω^{up}.

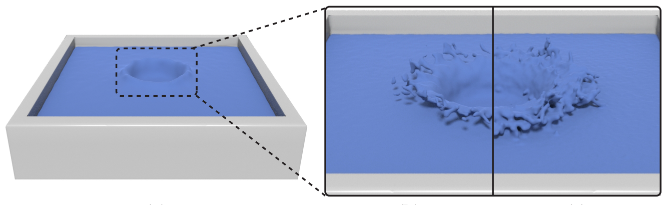

Fig. 9: *Spherical drop*: (a) Low-resolution (95k particles, 40^3 grid resolution, 0.9 s/frame) simulation of a spherical drop of liquid falling in a pool of liquid. (b) High-resolution (1.2M particles, 200^3 grid resolution, 7 s/frame) simulation. (c) Our method (115k added particles, 0.3 s/frame).

Volume Upsampling. In our implementation, we upsample Ω^S by using its discretized representation (i.e., set of cells). The input particles contained in the splash volume are automatically converted to upsampled particles. Therefore, we are able to prevent undesired artifacts that may occur by separately processing upsampled particles from the low-resolution input liquid. Using eight particles per grid cell is usually enough to extract fine details with the FLIP simulation model. As discussed by Zhu and Bridson [4], using more particles will rarely generate better results. However, since our approach aims to upsample an input coarse liquid for visual improvements, using locally more particles can greatly increase the small-scale details of the results. In fact, for the purpose of generating dynamic details for splashing liquids, we show that a higher density of particles will locally produce high-detailed and realistic results.

Seeding Tracers. As described by Bridson [53], we chose k to define the density of tracers for each sample x_i ($k = 5$ was used in all our examples). The samples x_i are initially defined at each center of the cells contained in the discretized narrow band of the liquid surface. Finally, the same distance as used with $\triangle\tau_{\text{tp}}$ is employed for the Poisson disk radius r. For effeciency, a tracer is discarded as soon as it touches the interface of the bulk liquid.

8. Results and Discussion

Our post-processing approach can locally generate realistic splashes in a wide range of scenarios. These splashes are obviously different than the ones that would result from a simulation with denser particles everywhere, but they are representative of such simulations since they affect both the splash details and

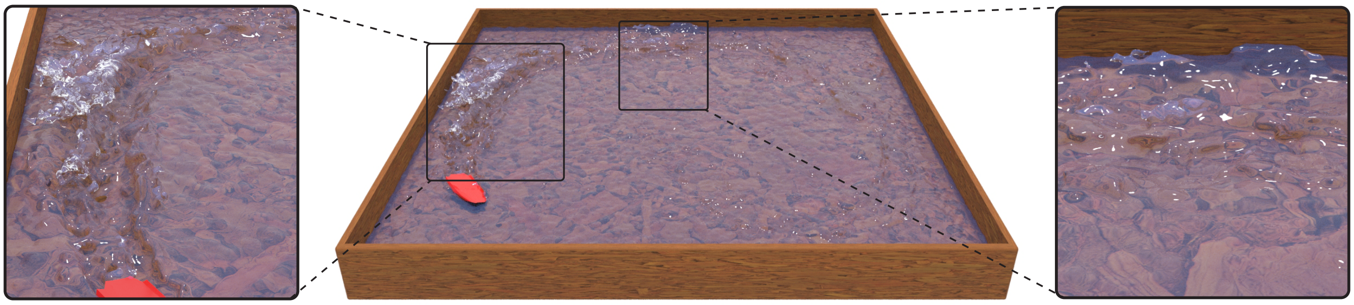
Fig. 10: Induced splash dynamics is generated on top of a coarse FLIP liquid provided as input. Our approach is able to reproduce localized realistic splashes (left enlarged) that were initially absent in the input coarse liquid. We also handle interactions between our geometric model and the coarse input, such as reflected swirls and waves (right enlarged), by adding an implicit linear wave model.

the coarse liquid surface. We can handle induced dynamics for solid-liquid as well as liquid-liquid interactions. In the following, we discuss the results obtained for the different examples presented in this paper and its supplemental material.

8.1. Comparison with High-Resolution Results

The idea behind our approach is to provide a sufficient density of particles locally to reduce the noticeable numerical dissipation occurring at sudden disturbance locations in the liquid. Our method focuses on generating splashes locally where they should appear. As shown in Fig. 9(a), the low-resolution liquid is unable to represent the small crown splash because of its very coarse discretization (95k particles). We had to increase the number of particles to slightly above a million (Fig. 9(b)) to successfully generate a fine-detailed splash with a traditional FLIP method.

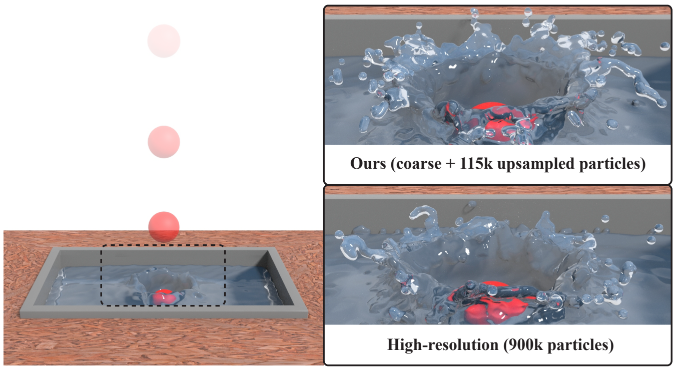
Ours (coarse + 115k upsampled particles)
High-resolution (900k particles)

Fig. 11: A ball falls in a small container of liquid. As clearly noticeable in the low-resolution example (left), the splash is totally nonexistent. Despite using this same coarse liquid as an input to our method, we managed to generate a detailed crown splash (top right), that is similar to the high-resolution reference (bottom right).

Despite the fact that the standard FLIP method provided enough fine details to extract a decent splash at high resolution, the computational cost of simulating this example was high (~7 s/frame) compared to the low-resolution version (~0.9 s/frame). Although this has been solved for hybrid methods, it is still worth mentioning that most of the high-resolution particles barely contributed to the desired result, even within a narrow band (e.g., [7, 8]). Our experiments on this example showed that only around 100k of these particles from the high-resolution simulation actually contributed to the central splash. With our post-processing approach, we need to add only 115k particles (based on criteria provided in § 4.2) on top of the coarse input particles, to generate a realistic splash reproducing a behavior matching the one found in the high-resolution simulation. Solely a small portion of the simulation domain needed to be processed by our method at a very inexpensive cost of about a second per frame from the coarse input liquid. Clearly, an arbitrary larger simulation domain would have resulted in larger gains.

8.2. Solid-Liquid Interactions

Our method has a fully automatic way to deal with solid-liquid interactions. The animated mesh of the solid interacting with the liquid is used to generate the associated splash volume. This step is computationally inexpensive and the frames only need to be processed once to compute and upsample the resulting splash volume. Another advantage of our approach is that once the pre-computation steps (splash volume and upsampling) are performed for a specific scenario, it can be used efficiently to experiment with different sets of values for parameters (as shown in Fig. 16).

Because our splash volume is computed by the interactions between a closed mesh and the input liquid, we must extend slightly the interface of the updated SDF by an epsilon value to gather the part of the coarse liquid in the vicinity of the moving object. For example, we used an epsilon equal to 0.2 (twice the particle separation) for the *Fractal fruits* (Fig. 1) and the *Ball drop* scenarios (Fig. 11) with a domain size of $8 \times 5 \times 8$. For the scenario of *Hulk's wrath* (Fig. 13), epsilon is 0.5 for dimensions of $10 \times 5 \times 3$. We extend the SDF to ensure that the resulting splash volume will contain most of the particles affected by this induced interaction. The *Speedboat ride* scenario (Fig. 10) is a good example to push the limits of our method. It would be reasonable to think that our method could be even more costly than a traditional FLIP method if the resulting splash volume were generated over the whole simulation domain. As shown in the *Speedboat ride* scenario, the path followed by the animated

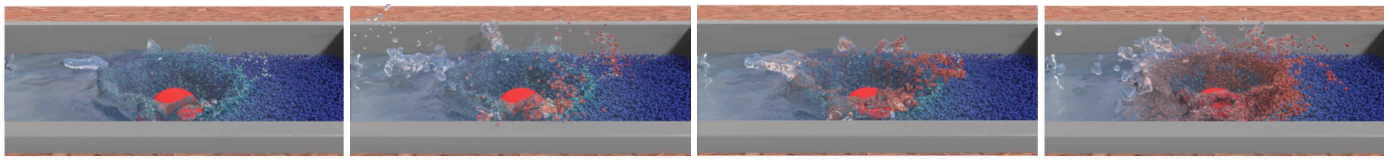

Fig. 12: Comparison between our approach and state-of-the-art methods using spray particles. The spray particles ((b) and (c)) and our upsampled particles (d) are identified in red.

Example	Number of particles			Pre-computation*		Simulation time (s/frame)			Simulation			
	Low-Res.	High-Res.	Upsampled	Splash Volume	Upsampling	LR	HR	Ours	Advection	Seeding Tracers	Artificial Pressure	Implicit Waves
Spherical drop	95k	1.2M	115k	2.3	1.4	0.9	7.0	0.3	14%	18%	23%	46%
Ball drop	100k	0.9M	115k	2.5	1.5	1.0	6.5	0.4	14%	16%	23%	47%
Hulk's wrath	225k	1.1M	150k	3.6	1.8	3.1	9.3	0.5	16%	23%	21%	40%
Speedboat ride	280k	2.3M	300k	34.2	2.5	3.5	10.5	0.7	17%	13%	19%	52%
Fractal fruits	400k	85.4M	450k	6.2	3.5	5.2	102.7	0.9	16%	14%	18%	52%
River	250k	10.6M	350k	3.3	5.2	2.6	11.2	0.8	15%	12%	17%	55%

Table 2: Statistics and computation times breakdown for the steps of our algorithm. Our method adds upsampled particles to the low-resolution particles. * The pre-computation steps (in seconds) are excluded from the simulation time (in seconds per frame) since they are computed once before simulating.

boat passes through most of the domain. Therefore, the evaluated splash volume at the end of the animation is indeed as large as the surface covering the whole liquid. However, even with a difficult scenario like this one, we managed to keep increasing the splash details while reducing significantly (compared to the equivalent high resolution) the associated computation time per frame. The criteria for upsampling are key to avoid oversampling at non-visually-interesting areas. As evidenced with the low-resolution example of this scenario, the boat does not generate much detail except when tight turns occur. From the input coarse resolution, our score based on the tendency of a particle to reach the surface helps to wisely upsample the underlying cells of the discretized splash volume. As shown in Table 2, the *Speedboat ride* example was more costly in pre-computation steps because we had to compute the volume based on the whole animation (500 frames). In contrast, the *Hulk's wrath* splash volume example was very fast to compute, considering that the hand was in contact with the liquid for just a few frames of the animation. In such cases, we were able to stop early the volume pre-computation step. The *Fractal fruits* (Fig. 1) example successfully demonstrates the capability and scalability of our post-processing approach. Once the required pre-computation steps are performed (total of 9.7 s to compute and upsample the splash volume for 200 frames), our method faithfully reproduces realistic and parameterizable splash details with less than one second per frame at runtime. According to this demonstration, our approach achieves a speedup factor of more than 100× over the high-resolution reference (see Table 2). Furthermore, the advantages of our approach also become apparent when simulating with different sets of values for parameters in order to obtain the desired result (e.g., as shown with Fig. 16). We also experiment with our approach in a more dynamic environment. The *River* scenario (available in the accompanying video) demonstrates the flexibility of our approach on a high-velocity flow. As explained in § 4.1, we compute the splash volume from the first collision between the upsampling and colliding volumes (i.e., Ω^{up} and Ω^{c}). However, in this case, since the initial velocity of the upsampling volume is fairly high, we process the upsampling step at the first collision frame.

	Fig. 9(a)	Fig. 9(b)	[42]	[40]	Ours
# spray / splash particles	-	-	49k	262k	115k
Time per iteration (ms)	900	7000	2	248	80

Table 3: Computation times comparison with state-of-the-art methods and ground truth (i.e., high resolution) on an example as shown in Fig. 12.

8.3. Liquid-Liquid Interactions

We also provide examples of liquid-liquid interactions. This type of scenario can produce an adequate splash volume by separating the input liquid into different volumes. For instance, with the *Spherical drop* scenario (Fig. 9), we divided at initialization the domain into two parts: the sphere and the liquid in the container. The sphere is handled similarly to a mesh in the sense that we pre-compute an SDF based on the related coarse particles. The resolution used for computing the level set needs to be high enough to capture and separate at the initial state (i.e., at the first frame or at emission) the input coarse liquid into distinct volumes. We used a resolution of 128^3 in all our examples and it has proved to be sufficient for our purposes.

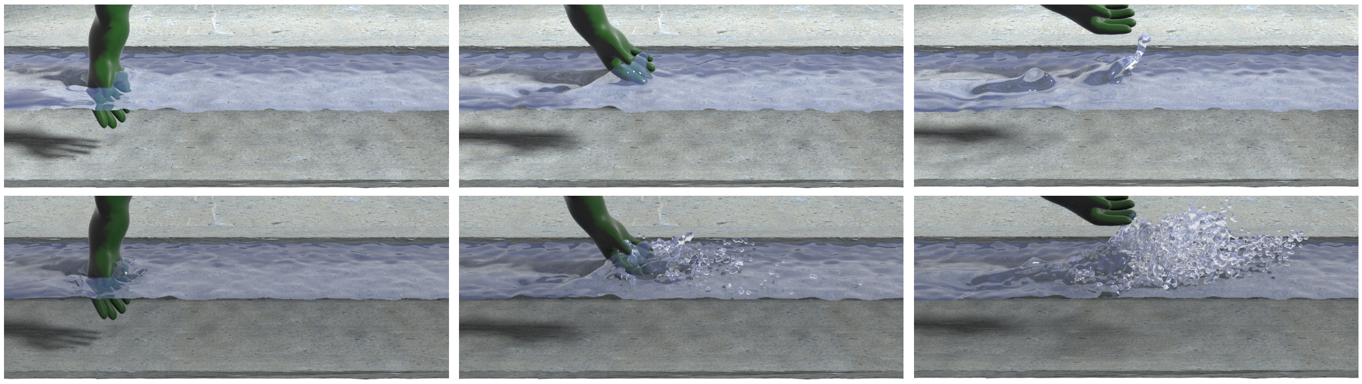

Fig. 13: *Hulk's wrath* is a good example of our approach with solid-liquid interactions. (top) The splash is almost nonexistent in the low-resolution example. (bottom) Our method manages to generate a realistic splashing liquid out of this very coarse liquid.

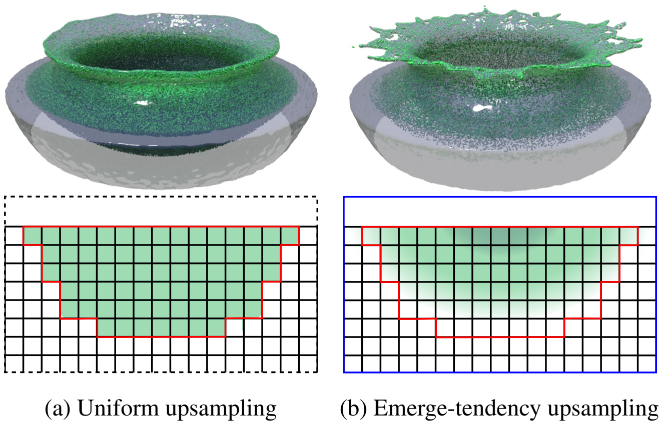

Fig. 14: This crown splash example shows a comparison with a naive (a) uniform upsampling method and (b) our upsampling method (right). (Bottom) The density of particles per cell is shown in green inside the splash volume (red).

8.4. Comparison with Spray Particles

We compared our approach with state-of-the-art spray particle methods. In order to present a fair comparison, only the contributions associated with spray particles are exposed. As shown in Fig. 12(b), Chentanez and Müller [42] introduce a very diffuse model to simulate spray particles. Since they target real-time applications, their spray model is added on top of a heightfield fluid, preventing particle-particle interactions. More similarly to ours, Ihmsen et al. [40] present a fully procedural spray model adapted for particle-based liquids. Their approach is capable of generating convincing diffuse behaviors, the motion of the spray particles is solely influenced by external forces and gravity (similar to our upsampled particles flagged as ballistics). This is where our approach and the purpose of tracers stand out. The localized forces applied by the tracers reproduce a better surface-tension behavior (Fig. 12(d)) as compared to the spray particles proposed by Ihmsen et al. [40] (Fig. 12(c)). Moreover, our implicit linear wave model generates smooth swirls at the base of the crown splash, introducing better interactions between our upsampled model and the coarse input liquid, as opposed to Ihmsen et al. [40].

8.5. Integration with Hybrid-DFSPH

To demonstrate that our approach works with most particle-based methods, we implemented our splash volume and upsampling method in a hybrid DFSPH model [8] (results are shown in the accompanying video). Although the velocity field can be estimated from the particle velocities, using the one provided from a hybrid model is even better. Our approach is completely independent of the input architecture. Nevertheless, we had to constraint the upsampling step to focus on regions with the splash volume bounded by the input narrow band of particles. We experiment with different strategies but doing so prevents us from generating artifacts deep under the liquid interface (i.e., in the Eulerian portion of the simulation).

8.6. Comparison with Uniform Upsampling

As shown in Fig. 14, our upsampling method faithfully preserves the small-scale details at the top of the crown splash (image on the right) while a uniform upsampling (image on the left) causes clustering of particles at the base of the splash, in regions where velocity magnitudes tend to be zero over time. Adding particles to these regions will hardly contribute to increase details. Our upsampling method offers a more effective way to upsample and distribute particles within the splash volume. As shown in the graph of Fig. 15, our upsampling method presents a distribution trend similar to the one of the Maxwell-Boltzmann distribution. As stated by Mandl [59] and in reference to the Maxwell-Boltzmann statistics, the density of particles highly depends on the distribution of velocities inside the fluid. By oversampling particles in the regions identified by our method, we ensure to solely increase particle density to bring out fine details dissipated and bounded by the input resolution. Also, since our method highly upsample a subdomain of the input simulation, preserving such a distribution is critical for detail enhancement and to avoid undesired particle clumping. As expected, our upsampling method preserves a high density of particles in cells with velocities close to $\|\bar{\vec{u}}\|$ (average velocity norm). Low densities of particles are kept in cells with low velocities and very high velocities (way above average). The low densities of particles in high-velocity cells correspond to highly diffuse particles (e.g., such as small droplets).

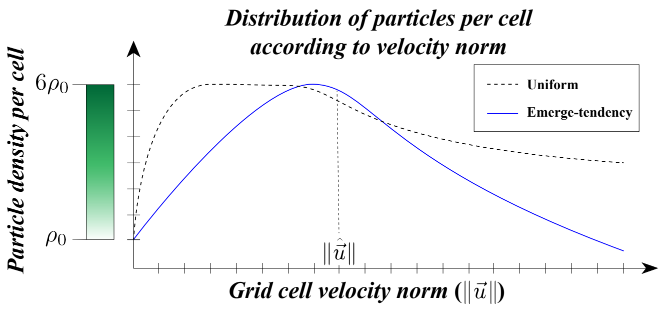

Fig. 15: This graph plots the density against the velocity distribution exclusively within the splash volume of the scenario shown in Fig. 14. ρ_0 corresponds to the number of particles per cell as used by the input coarse liquid; $6\rho_0$ represents the maximum number of upsampled particles used in this examples (i.e., $s_{up} = 6$). Also note that the dash plot (uniform upsampling) indicates many more particles in total (upsampled) than the continuous blue plot (emerge-tendency upsampling).

9. Conclusion and Future Work

We have presented a flexible and physics-motivated post-processing approach to generate detailed and realistic splash behaviors from a coarser particle-based input liquid. Our approach is divided into different steps. A splash volume is computed and upsampled by analyzing the surrounding environment and by determining where sudden disturbances occur. The upsampled particles advected alongside the coarse liquid provide a practical way to enrich portions of liquid that could not otherwise capture such details. We also provided a novel way to affect splashing particles through tracers and localized artificial pressure corrections. Finally, our efficient implicit wave model offers a controllable method to process interactions between upsampled splash particles and input particles. As noted by the statistics presented in Table 2, even if we compare the computation times by adding the simulation of a low-resolution liquid to our upsampling method, we still manage to outperform the reference high-resolution simulations. In addition, once the pre-calculation steps are completed, our method allows a user, without any limitation, to adjust the level of detail almost interactively.

Nevertheless, our method suffers from a few limitations. First of all, precomputing the splash volume once for the whole animation prevents us from upsampling at different time intervals. In other words, highly dynamic scenarios would require an adaptive upsampling method to address frequent changes in the regions of interest through time. Our approach is currently designed to better suit scenarios in which the upsampled portion is initially static. An evolving 4D splash volume would be an interesting direction for future work. By tracking these splash volumes temporally, we could improve our upsampling method to cover complex interactions. Secondly and along the same lines, while our method offers a controllable way to improve the apparent resolution of splashing liquids, some high-resolution scenarios would hardly benefit from its current state. For example, simulating the collision between a high-velocity volume of liquid and a complex geometry (e.g., firehose shooting water on a highly detailed armadillo) would barely benefit from our proposed upsampling approach.

Because of the flexibility of our approach, we plan to explore more art-directable methods to use controls similar to our tracers as a modeling tool for splashes and turbulent flows. It would be also relevant to improve the steps of upsampling and tracer seeding at different scales through machine learning techniques. State-of-the-art neural network approaches could improve the prediction on seeding locations for better results. It could also provide an efficient way to upsample particles based on learned high-resolution examples, and result in even more similar splashes.

References

[1] Solenthaler, B, Pajarola, R. Predictive-corrective incompressible SPH. ACM Trans on Graphics (TOG) 2009;28(3):40.
[2] Ihmsen, M, Cornelis, J, Solenthaler, B, Horvath, C, Teschner, M. Implicit incompressible SPH. IEEE Trans on Visualization and Computer Graphics 2014;20(3):426–435.
[3] Bender, J, Koschier, D. Divergence-free SPH for incompressible and viscous fluids. IEEE Trans on Visualization and Computer Graphics 2017;23(3):1193–1206.
[4] Zhu, Y, Bridson, R. Animating sand as a fluid. ACM Trans on Graphics (TOG) 2005;24(3):965–972.
[5] Boyd, L, Bridson, R. MultiFLIP for energetic two-phase fluid simulation. ACM Trans on Graphics (TOG) 2012;31(2):16.
[6] Ando, R, Thurey, N, Tsuruno, R. Preserving fluid sheets with adaptively sampled anisotropic particles. IEEE Trans on Visualization and Computer Graphics 2012;18(8):1202–1214.
[7] Ferstl, F, Ando, R, Wojtan, C, Westermann, R, Thuerey, N. Narrow band FLIP for liquid simulations. Computer Graphics Forum 2016;35(2):225–232.
[8] Roy, B, Poulin, P. A hybrid Eulerian-DFSPH scheme for efficient surface band liquid simulation. Computers & Graphics 2018;77:194–204.
[9] Ihmsen, M, Orthmann, J, Solenthaler, B, Kolb, A, Teschner, M. SPH fluids in computer graphics. In: Eurographics - State of the Art Reports. 2014,.
[10] Bridson, R. Fluid Simulation for Computer Graphics. CRC Press; 2015.
[11] Kim, T, Thürey, N, James, D, Gross, M. Wavelet turbulence for fluid simulation. ACM Trans on Graphics (TOG) 2008;27(3):50.
[12] Narain, R, Sewall, J, Carlson, M, Lin, MC. Fast animation of turbulence using energy transport and procedural synthesis. ACM Trans on Graphics (TOG) 2008;27(5):166.
[13] Schechter, H, Bridson, R. Evolving sub-grid turbulence for smoke animation. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2008, p. 1–7.
[14] Pfaff, T, Thuerey, N, Selle, A, Gross, M. Synthetic turbulence using artificial boundary layers. ACM Trans on Graphics (TOG) 2009;28(5):121.
[15] Yuan, Z, Zhao, Y, Chen, F. Incorporating stochastic turbulence in particle-based fluid simulation. The Visual Computer 2012;28(5):435–444.
[16] Shao, X, Zhou, Z, Zhang, J, Wu, W. Realistic and stable simulation of turbulent details behind objects in smoothed-particle hydrodynamics fluids. Computer Animation and Virtual Worlds 2015;26(1):79–94.
[17] Selle, A, Rasmussen, N, Fedkiw, R. A vortex particle method for smoke, water and explosions. ACM Trans on Graphics (TOG) 2005;24(3):910–914.
[18] Solenthaler, B, Gross, M. Two-scale particle simulation. ACM Trans on Graphics (TOG) 2011;30(4):81.
[19] Ando, R, Tsuruno, R. A particle-based method for preserving fluid sheets. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2011, p. 7–16.
[20] Ando, R, Thürey, N, Wojtan, C. Highly adaptive liquid simulations on tetrahedral meshes. ACM Trans on Graphics (TOG) 2013;32(4):103.
[21] Kim, T, Tessendorf, J, Thuerey, N. Closest point turbulence for liquid surfaces. ACM Trans on Graphics (TOG) 2013;32(2):15.
[22] Mercier, O, Beauchemin, C, Thuerey, N, Kim, T, Nowrouzezahrai, D. Surface turbulence for particle-based liquid simulations. ACM Trans on Graphics (TOG) 2015;34(6):202.

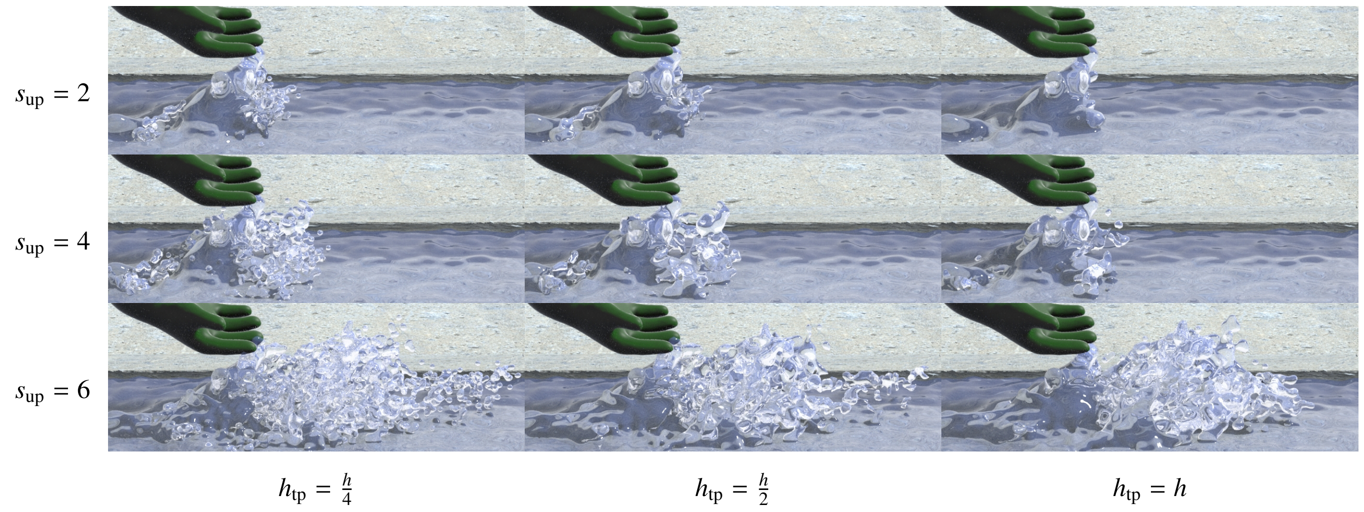

Fig. 16: With the *Hulk's wrath* example, we show the influence of the scaling factor s_{up} on the number of upsampled particles per cell (rows) and the tracers support radius h_{tp} (columns) on the appearance of the liquid.

[23] Thuerey, N, Kim, T, Pfaff, T. Turbulent fluids. In: ACM SIGGRAPH 2013 Courses. 2013, p. 6.

[24] Desbrun, M, Cani, MP. Space-time adaptive simulation of highly deformable substances. Ph.D. thesis; INRIA; 1999.

[25] Adams, B, Pauly, M, Keiser, R, Guibas, LJ. Adaptively sampled particle fluids. In: ACM Trans on Graphics (TOG); vol. 26. 2007, p. 48.

[26] Orthmann, J, Kolb, A. Temporal blending for adaptive SPH. In: Computer Graphics Forum; vol. 31. 2012, p. 2436–2449.

[27] Horvath, CJ, Solenthaler, B. Mass preserving multi-scale SPH. Tech. Rep. 13–04; Pixar Animation Studios; 2013.

[28] Zhu, B, Lu, W, Cong, M, Kim, B, Fedkiw, R. A new grid structure for domain extension. ACM Trans on Graphics (TOG) 2013;32(4):63.

[29] Aanjaneya, M, Gao, M, Liu, H, Batty, C, Sifakis, E. Power diagrams and sparse paged grids for high resolution adaptive liquids. ACM Trans on Graphics (TOG) 2017;36(4):140.

[30] Sato, T, Wojtan, C, Thuerey, N, Igarashi, T, Ando, R. Extended narrow band FLIP for liquid simulations. In: Computer Graphics Forum; vol. 37. 2018, p. 169–177.

[31] Ibayashi, H, Wojtan, C, Thuerey, N, Igarashi, T, Ando, R. Simulating liquids on dynamically warping grids. IEEE Trans on Visualization and Computer Graphics 2018;.

[32] Ladicky, L, Jeong, S, Solenthaler, B, Pollefeys, M, Gross, M. Data-driven fluid simulations using regression forests. ACM Trans on Graphics (TOG) 2015;34(6):199.

[33] Chu, M, Thuerey, N. Data-driven synthesis of smoke flows with CNN-based feature descriptors. ACM Trans on Graphics (TOG) 2017;36(4):69.

[34] Um, K, Hu, X, Thuerey, N. Liquid splash modeling with neural networks. Computer Graphics Forum 2018;37(8):171–182.

[35] Xiao, X, Yang, C, Yang, X. Adaptive learning-based projection method for smoke simulation. Computer Animation and Virtual Worlds 2018;29(3-4):e1837.

[36] Takahashi, T, Fujii, H, Kunimatsu, A, Hiwada, K, Saito, T, Tanaka, K, et al. Realistic animation of fluid with splash and foam. Computer Graphics Forum 2003;22(3):391–400.

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

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

[39] Wang, CB, Zhang, Q, Kong, FL, Qin, H. Hybrid particle-grid fluid animation with enhanced details. The Visual Computer 2013;29(9):937–947.

[40] Ihmsen, M, Akinci, N, Akinci, G, Teschner, M. Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer 2012;28(6-8):669–677.

[41] Kim, J, Cha, D, Chang, B, Koo, B, Ihm, I. Practical animation of turbulent splashing water. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2006, p. 335–344.

[42] Chentanez, N, Müller, M. Real-time simulation of large bodies of water with small scale details. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2010, p. 197–206.

[43] Um, K, Hu, X, Thuerey, N. Perceptual evaluation of liquid simulation methods. ACM Trans on Graphics (TOG) 2017;36(4):143.

[44] Kass, M, Miller, G. Rapid, stable fluid dynamics for computer graphics. In: ACM SIGGRAPH Computer Graphics; vol. 24. 1990, p. 49–57.

[45] Yuksel, C, House, DH, Keyser, J. Wave particles. ACM Trans on Graphics (TOG) 2007;26(3):99.

[46] Tessendorf, J. Vertical derivative math for iWave. Tech. Rep.; Clemson University; 2008.

[47] Cords, H. Moving with the flow: Wave particles in flowing liquids. Journal of WSCG 2008;:145–152.

[48] Wang, H, Miller, G, Turk, G. Solving general shallow wave equations on surfaces. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2007, p. 229–238.

[49] Yang, S, He, X, Wang, H, Li, S, Wang, G, Wu, E, et al. Enriching SPH simulation by approximate capillary waves. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2016, p. 29–36.

[50] Canabal, JA, Miraut, D, Thuerey, N, Kim, T, Portilla, J, Otaduy, MA. Dispersion kernels for water wave simulation. ACM Trans on Graphics (TOG) 2016;35(6):202.

[51] Jeschke, S, Wojtan, C. Water wave packets. ACM Trans on Graphics (TOG) 2017;36(4):103.

[52] Macklin, M, Müller, M. Position based fluids. ACM Trans on Graphics (TOG) 2013;32(4):104.

[53] Bridson, R. Fast Poisson disk sampling in arbitrary dimensions. In: SIGGRAPH sketches. 2007, p. 22.

[54] Monaghan, JJ. SPH without a tensile instability. Journal of Computational Physics 2000;159(2):290–311.

[55] Solenthaler, B, Pajarola, R. Density contrast SPH interfaces. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2008, p. 211–218.

[56] Müller, M, Charypar, D, Gross, M. Particle-based fluid simulation for interactive applications. In: ACM SIGGRAPH/Eurographics Symp. on Computer Animation. 2003, p. 154–159.

[57] Müller-Fischer, M. Fast water simulation for games using height fields. In: Proc. of the Game Developer's Conference. 2008, p. 24.

[58] SideFX, . Houdini (version 16.5). `https://www.sidefx.com/products/houdini-fx/`; 2016.

[59] Mandl, F. Statistical Physics (2nd Edition). John Wiley & Sons; 2008.