Volume xx (*200y*), *Number z, pp. 1–12*

Temporal Blending for Adaptive SPH

paper1390

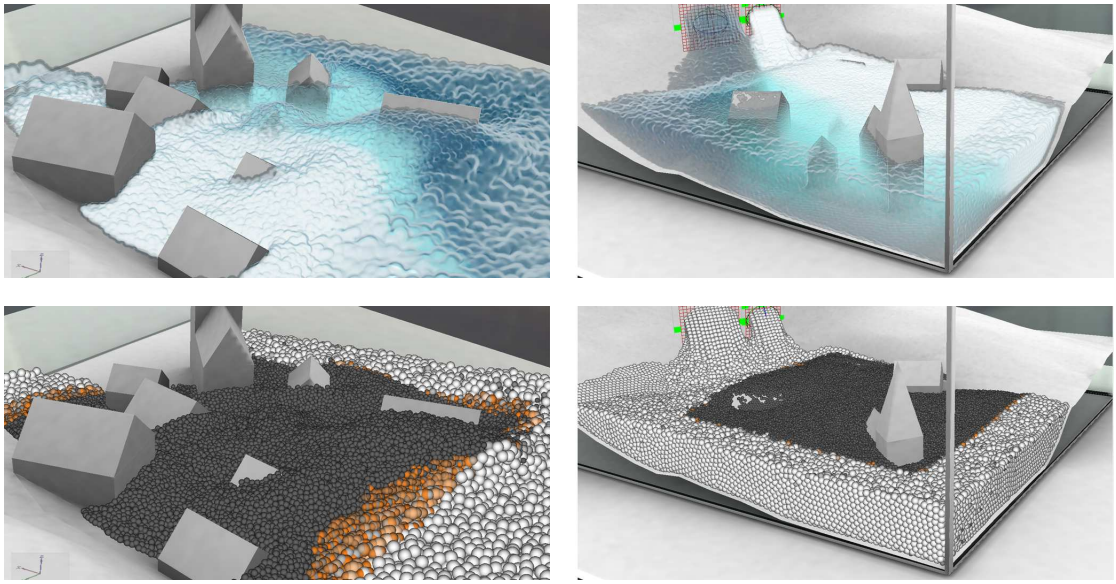


Figure 1: *Two fluids flooding a valley with a salt diffusion (from white to blue) in our interactive fully GPU-based PCISPH shown for two different time-steps. The particle resolution is smoothly doubled around the village by using blend-sets (orange).*

Abstract

In this paper we introduce a fast and consistent Smoothed Particle Hydrodynamics (SPH) technique which is suitable for convection-diffusion simulations. We employed a temporal blending technique to reduce the number of particles in the simulation while smoothly changing quantity fields. Our approach preserves the accuracy and minimizes the error in the pressure term introduced when changing particle configurations. Compared to other methods, this allows to apply larger time-steps in the transition phase. Our implementation is fully GPU-based in order to take advantage of the parallel nature of SPH simulations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The simulation of fluids is important to many applications such as physically based animation and interactive computer graphics. Compared to grid-based (Eulerian) techniques, particle-based (Lagrangian) approaches, like Smoothed Particle Hydrodynamics (SPH) automatically include conservation of mass and are very suitable to handle free surfaces. SPH, as introduced by Gingold and Monaghan [GM77], models the dynamics of fluids based on particle motions applying forces to ensure the Navier-Stokes equations. Various attempts have been made to optimize SPH by improving the

accuracy and/or reducing the computational costs. Regarding accuracy, SPH for incompressible fluids has been introduced to Computer Graphics by Becker et al. [BT07]. Different SPH parameters are crucial for the performance of a SPH simulation for incompressible fluids: for example, using small particle support radii [MCG03] minimizes the neighbourhood complexity. Dynamically adjusting the integration time step based on the Courant-Friedrich-Levy (CFL) condition shortens the overall simulation time [IAGT10]. Another intuitive idea to speed-up simulations is to reduce the overall particle count by using adaptive particle sizes [APKG07] or two coexisting resolution levels [SG11]. Beside optimizing SPH parameters, prediction-correction steps enable larger integration time-steps [SP09] for incompressible fluids (PCISPH). Last but not least, graphics processing units (GPUs) are able to exploit the highly parallel nature of SPH simulations [GSSP10].

The overall goal of this paper is to provide a consistent and an accurate adaptive SPH while allowing for large integration step sizes. For this purpose, we introduce a feature-specific adaptation of the influencing SPH-parameters, i.e. the number of particles in combination with an adaptive time stepping and small support radii using prediction-correction steps. However, an instantaneous replacement of particle configurations introduces a significant change in the pressure term. This leads to small time steps when CFL conditions are dynamically enforced. Our main idea is to apply a temporal blending scheme to stabilize the error in the pressure term. In detail, our approach incorporates the following contributions:

- a novel approach for a consistent adaptive SPH using a temporal blending of quantities which allows to use large time-steps and,
- a scheme to estimate the blending step size based on a predicted error in the pressure term enabling an error-dependent blending time, and
- a solely GPU-based implementation for incompressible fluids including an efficient handling of explicit neighbour lists for each particle exploiting the highly parallel nature of today's GPUs.

Compared to prior approaches, the temporal blending proves to be very robust in terms of the pressure error, allowing for large integration time-steps. As such, we believe that it is possible to integrate the proposed temporal blending into any other existing SPH scenario where a smooth exchange of particle sets is required which might not only be motivated by performance reasons.

In the remainder of the paper, we first discuss related work (Sec. 2) and introduce relevant aspects of our adaptive SPH (Sec. 3) including a motivation for our temporal blending approach as presented in Sec. 4. Implementation aspects are given in Sec. 5. In Sec. 6 we present advantages and limitation of our adaptive re-sampling which we conclude in Sec. 7.

2. Related Work

Since the introduction of SPH by Monaghan [Mon92, Mon05], many improvements on computation speed have been introduced. We cluster them according to the complexity of the underlying physics, data-parallelism in combination with neighbor-finding and the adaption of particle sets. For further details, we refer the reader to surveys, e.g. [KCR08, Bri08].

Physics: Desbrun et al. [DC96] introduced SPH to the computer graphics community. They designed kernels with small compact support radii in order to decrease the total number of particle interactions. Based on this work, Müller et al. [MCG03] presented a set of smoothing kernels with compact support in order to simulate fluids at interactive rates. Becker et al. [BT07] employed Tait's equation of state to enforce incompressibility, however, requiring small timesteps. Instead, Solenthaler et al. [SP09] employed a predictive-corrective loop with a small number of iterations allowing for larger integration-time steps. Ihmsen et al. [IAGT10] extended their approach by smoothly adapting the integration time for all particles after a simulation step. Additionally, they have improved direct forcing at boundaries as proposed by Becker et al. [BTT09] by including boundary particles in the pressure update loop. Beside convective flux, diffusion for SPH has been introduced to computer graphics by Stora et al. [SAC*99] to animate lava flows which has been extended by Müller et al. [MSKG05] to simulate a fluid-fluid interaction. However, for SPH the laplacian is better approximated by using an integral approximation [CM99] as used by Kristof et al. [KBKS09] to simulate the transport of sediments.

Data-Parallelism and Neighbours: Kolb et al. [KC05] and Kipfer et al. [KSW04] were the first who took advantage of the data parallel nature of particle systems on programmable graphics hardware. In contrast, collecting quantities is to be preferred on parallel architectures as it reduces costly memory collisions [SDG08]. The first gathering approach on the GPU has been published by Harada et al. [HKK07], incorporating a bucket structure on the GPU to accelerate neighborhood searches. Green [Gre09] and Le Grand et al. [Gra08] have taken this idea one step further by sorting particles according to their current location in a fixed size hash-grid. Ihmsen et al. [IABT11] and Goswami et al. [GSSP10] improved their cache-coherence by utilizing a z-indexing. Furthermore, Goswami et al. [GSSP10] split bins according to shared memory demands on the GPU.Pelfrey et al. [PH10], however, maintain neighbour references for each particle in order to avoid unnecessary memory reads which at the same time simplify SPH operations.

Adaptivity: Reducing the overall particle count either locally or globally is very appealing in order to improve the simulation efficiency as for SPH, the overall computational cost increases linearly with the number of particles. Bonet et

al. [FB07] utilize a non-linear minimization of the density error. However, their method is too time critical for interactive simulations. In computer graphics simple replacement schemes are used to allow for a high performance, however, increasing the approximation error.[DC99, ZSP08] employed the differential form of the continuity equation in order to avoid high local changes of mass-density due to split or merge operations. Unfortunately, in the differential form, the error introduced into the density field is accumulated over time which leads to instabilities. As first published by Becker et al. [BT07] the summation approach is much more stable for large time steps, as was confirmed by our experiments as well. Lastiwka et al. [LQB05] relate a particle's support radius to the estimated volume after re-sampling using large support radii, however, inducing additional computational costs. Zhang et al. [ZSP08] adjust masses to stabilize the differential continuity equation, violating the principle of mass conservation. Adams et al. [APKG07] determine valid positions for splitted particles and stop if they are too close to their neighbours in order to avoid instantaneous pressure changes. Keiser et al. [KAG*06] have utilized virtual particles, i.e. particle which are passively moved with the flow. As virtual particles do not get a direct contribution from real particles they introduce errors to quantity fields when they turn into real particles. In order to avoid merging and splitting Cottet et al. [CKS00] divide the fluid domain into several areas with different resolutions. Unfortunately, their system depends on a frequent global remeshing of all particles onto a regular grid [CPK02], which distracts from the adaptive character of Lagrangian systems. Recently, Solenthaler et al. [SG11] have proposed a particle simulation with coexisting resolution levels with the nice property to quadruple resolution. Unfortunatelly, the utilized feedback forces are not physically motivated and do not prevent divergence between resolution levels, which may lead to conservation problems especially in case of unbounded free surface scenarios or diffusion simulations, such as shown in Fig.1. In contrast, in our approach particle levels interact using standard SPH rules enabling multi-level convection-diffusion simulations.

3. Adaptive SPH

In this section, we describe SPH basics (Sec. 3.1) for convection-diffusion simulations and the principles of adaptive SPH systems (Sec. 3.2) such as shown in Fig. 2. In Sec. 3.3 we discuss major challenges in realizing an adaptive sampling mechanism while giving reason for our proposed blending of quantities as described in Sec.4. Our specific implementation is described later in Sec. 5.

3.1. SPH Background

In Lagrangian systems, particles represent mass points which are moved with the flow field. Beside a mass value m, particles carry specific flow properties. In SPH, such a quantity is reconstructed at a position $\mathbf{x}$ by particles $\mathbf{x}_j$ in the neighbourhood:

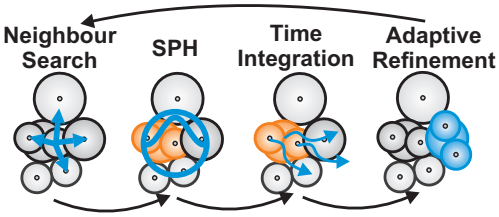


Figure 2: *A schematic overview of a typical simulation loop for adaptive SPH systems. Our temporal blending easily integrates into SPH field evaluations and uses an error estimation during time-integration in order to avoid artificial shocks caused by an adaptive particle sampling.*

$$Q(\mathbf{x}) = \sum_j Q_j \frac{m_j}{\rho_j} W(|\mathbf{x}-\mathbf{x}_j|, h_j), \tag{1}$$

where $W(|\mathbf{x}-\mathbf{x}_j|, h_j)$ is a radial symmetric kernel function with small compact support radius h_j [MCG03]. We use the short notation $W_{ij} = W(|\mathbf{x}_i - \mathbf{x}_j|, h_j)$ in case it is evaluated for a particle i. Mass transport of a soluble substance is described by convection-diffusion equations of which the velocity $\mathbf{v}_i$ is described by the Navier-Stokes equations:

$$\frac{\partial \mathbf{v}_i}{\partial t} = \frac{1}{m_i}[\mathbf{F}_i^p + \mathbf{F}_i^\mu + \mathbf{F}_i^\sigma + \mathbf{F}_i^e].$$

$\mathbf{F}_i^e$ is an external force, e.g. gravity. For the viscous force $\mathbf{F}_i^\mu$ and the surface tension force $\mathbf{F}_i^\sigma$ we refer to [Mon05] and [BT07], respectively. However, a particle's pressure force $\mathbf{F}_i^p$ is derived from symmetrized gradient approximations as described by [CEL06]:

$$\mathbf{F}_i^p = -m_i \sum_{j \neq i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij} \tag{2}$$

where p_i and ρ_i are the pressure and the density of a particle. In summation form, the density $\rho_i = \rho(\mathbf{x}_i)$ for a particle is computed by:

$$\rho_i = \sum_j m_j W_{ij}. \tag{3}$$

Pressure is modelled via an equation of state and e.g. may linearly depend on the density via $p = k(\rho - \rho_0)$, where ρ_0 e.g. is the rest density of a fluid. k is either defined by the compression ratio [MCG03] or is precomputed over an optimal neighbourhood for incompressible fluids [SP09]. Beside convection, an isotropic diffusion is modelled by a exchange of concentrations c of a soluble substance, like salt in Fig.1, between neighbouring particles [CM99]:

$$\frac{\partial c_i}{\partial t} = D \sum_{j \neq i} \frac{m_j}{\rho_j \rho_i} (c_i - c_j) |\nabla W_{ij}|. \tag{4}$$

D is the diffusivity constant, controlling how fast substances propagate in fluids.

3.2. Adaptive Sampling

Figure 3: *Mixing of coffee and cream in an Utah Teapot. Blend-sets (bottom) are used to dynamically adapt to the convective and the diffusive flux (top), shown for t=2 and t=6.*

An adaptive mechanism shifts computational resources to regions of interest. Adaptive simulations use particles with non-uniform support radii h_i and masses m_i depending on their current level $l = 0, 1, 2, ..., l_{\max}$:

$$h_i = \sigma \left(\frac{m_i}{\rho_0} \right)^{\frac{1}{3}}, \quad m_i = 2^l m_0 \tag{5}$$

where ρ_0 is the rest density of the fluid, m_0 is the reference mass for level zero particles. Here $\sigma \approx 1.3$ in order to conserve a fluid's volume [Mon05] over resolution levels.

High-resolutions are either pre-defined (See Fig. 1) or change dynamically, as dictated by the flow. For our examples, such an adaptive high-resolution region is defined by the fluid's surface in combination with the diffusion layer, i.e. the contact-line between the two fluids as shown in Fig. 3. Particles $\mathbf{x}_i$ belong to a high-resolution area as long as one of the following conditions holds true:

$$|\sum_j \frac{m_j}{\rho_j} \nabla W_{ij}| > \varepsilon_s \quad \text{or} \quad |\sum_j \frac{m_j}{\rho_j} (c_i - c_j) \nabla W_{ij}| > \varepsilon_c$$

where $\varepsilon_s, \varepsilon_c$ are certain user defined thresholds for the volume gradient (See also Mueller et al. [MSKG05]) and the concentration gradient, respectively. Once high-resolution particles are identified, simple sampling operators, such as shown in Fig.5, are used to refine a local particle set, aiming for a good performance. High-resolution particles of level l try to split to N child particles of level $l - \log_2(N)$. Vice versa all other particles, not in a high-resolution region, try to merge to a single particle of level $l + 1$. Similar to Adams et al. [APKG07], we leave an intermediate area of particles which are not re-sampled in order to avoid split-merge fluctuations. However, many other re-sampling criteria exist throughout the literature [SZP07]. Please note that field quantities, like velocities and concentrations, are not directly assigned and are instead interpolated over time as described later in Sec. 4.3.

However, consistent adaptive systems differ in the way particles of different levels contribute to each other. Particles may only contribute to particles of their level [KAG*06] or may exchange information with all neighbour particles directly [APKG07]. In the latter case, which we use due to it's simplicity, according to Monaghan [Mon92] one has to average the influence of neighbour particles in order to symmetrize contributions and satisfy Newton's third law for action and reaction:

$$W_{ij} = W(|\mathbf{x}_i - \mathbf{x}_j|, \frac{h_i + h_j}{2}),$$

Please note that alternative averaging operators [DC99] may be applied as well.

Due to smaller particle sizes, high resolution regions need smaller integration time-steps in order to secure simulation stability. According to the Courant-Friedrich-Levy (CFL) condition the velocity $\mathbf{v}_i$, the force F_i and the support radii h_i of a particle i then determine it's maximum allowed time-step:

$$\Delta t_i = \min(\lambda_v \frac{h_i}{|\mathbf{v}_i|}, \ \lambda_F \sqrt{\frac{h_i}{|\mathbf{F}_i|}}). \tag{6}$$

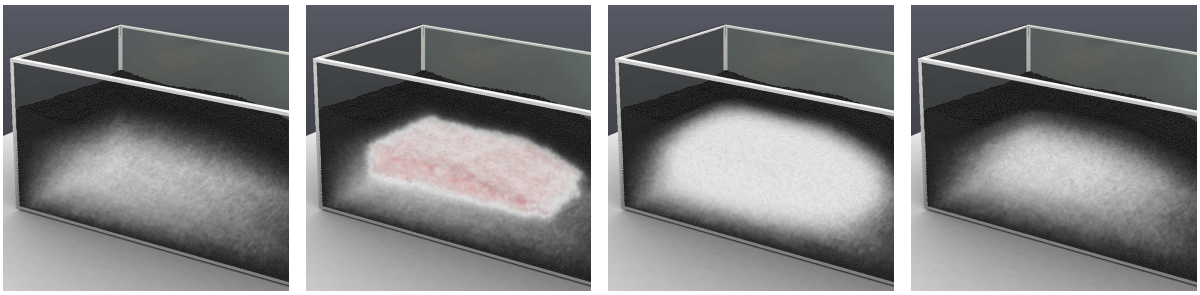
where $\lambda_v = 0.4$ and $\lambda_F = 0.25$ according to Monaghan [Mon92]. The overall simulation speed then depends on the minimum over all individual time-steps. However, in the context of PCISPH the time-step must be changed smoothly and requires a separate shock handling as described by Ihmsen et al. [IAGT10].

3.3. Challenges for Adaptive SPH

Adaptive SPH systems re-sample particles globally or locally, while preserving simulation accuracy. Consequently, a sampling mechanism has to approximate a particle set as good as possible. In case of regular particle lattices [CPK02], a higher order interpolation in combination with frequent global remeshing is applied which distracts from the adaptive character of particle sets. Instead, in case of free surface flows including irregular particle structures, one has to minimize a local error function of a quantity field at position $\mathbf{x}$ [FB07]:

$$E(\mathbf{x}) = |Q(\mathbf{x}) - \widetilde{Q}(\mathbf{x})|. \tag{7}$$

Here, $Q(\mathbf{x})$ is the quantity evaluated for the old particle set and $\widetilde{Q}(\mathbf{x})$ is the quantity interpolated on the re-sampled particle field. Lagrange multipliers and costly iterative solvers are required in order to conserve the total amount of quantities and to account for non-negativity constraints [LB95]. In computer graphics, simple sampling patterns as described



(a) Ground Truth (b) Non Continuous (c) Error-Independent (d) Error-Dependent

Figure 4: *In 4(a) we visualize the pressure distribution for the scene shown in Fig.8 after 1.5 seconds of simulated real-time. In 4(b) high pressure is introduced due to an abrupt merging of 60k particles as utilized by [APKG07, KAG*06] in the context of PCISPH leading to an unstable simulation in case the number of merging particles is not reduced as shown later in 8. In 4(c) our error-independent linear blending function is applied which gives a result close to the original pressure field. However, an error estimation in combination with our linear blending preserves the overall pressure distribution much better as shown in 4(d).*

in Sec. 3.2 distribute flow quantities equally to the new child particle(s) which are within the same distance to their parent particle(s). In general, such operators do not include any neighbour particles to minimize $E(\mathbf{x})$ and cannot preserve the overall regular particle structure resulting in possible particle overlaps. Thus, they are very fast to compute but approximate the old quantity field with larger errors. Fig. 4 gives an example of the resulting error which is introduced into the density field. Differentials in SPH are quite sensitive to such irregular particle structures and field discontinuities. Thus, they are sensitive to abrupt changes in the particle set. Solenthaler et al. [SG11] have utilized an impulse-based transition for high-level boundary particles turning into real particles. It is applicable only if two resolution levels are allowed to coexist within the same region, possibly leading to divergence problems. In consistent systems communication with different smoothing radii increases the error as well, as shown by Borve et al. [BOT01]. In general, this error can be reduced by avoiding direct communication among levels as proposed by Keiser et al. [KAG*06] or by using a $1:2$ replacement structure as has been proposed by Adams et al. [APKG07]. In either case, large pressure forces are introduced, as a result of a non-optimized sampling, irregular particle distributions, different support radii and small compact smoothing kernels. By applying the CFL condition in each step, such forces dramatically decrease the integration time in order to preserve simulation stability and accuracy. Even worse, in the context of PCISPH such forces may trigger a shock handling mechanism as described by Ihmsen et al. [IAGT10]. Instead, our blending approach reduces the error of irregular particle structures for consistent simulations by using an error-dependent (Sec. 4.4) transition over time via our adaptive quantity blending (Sec. 4.1- 4.3). Thus, we enable the system to use large time steps in combination with small smoothing kernels which are required for a fast and incompressible SPH.

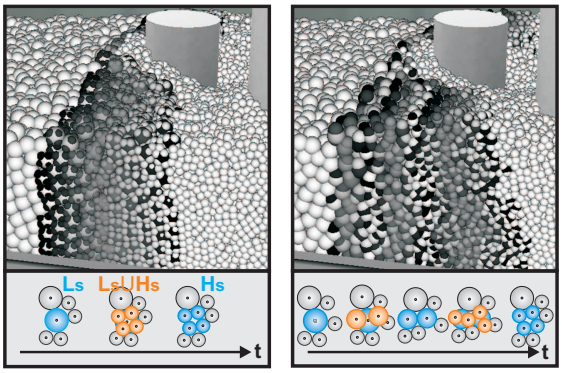
4. Temporal Blending

As shown in Fig. 4, sampling errors for new particles lead to discontinuous quantity fields over time. Instead, we propose a blending approach in order to smoothly interpolate between two interchangeable fluid representations over time. The idea of blending between multiple representation of objects is well known in computer-graphics.The following summation transfers the idea of a blending between two representations of a fluid into the context of SPH:

$$Q(\mathbf{x}) = \overbrace{b \sum_{j \in H} Q_j \frac{m_j}{\rho_j} W(|\mathbf{x}-\mathbf{x}_j|)}^{Q_H(\mathbf{x})} + \underbrace{(1-b) \sum_{j \in L} Q_j \frac{m_j}{\rho_j} W(|\mathbf{x}-\mathbf{x}_j|)}_{Q_L(\mathbf{x})} . \quad (8)$$

Here, a low-resolution particle set L and a high-resolution particle set H, represent two interchangeable **blend-domains**. Both corresponding quantity fields $Q_H(\mathbf{x})$ and $Q_L(\mathbf{x})$ are smoothly blended with respect to a **blend-weight** $b \in [0,1]$. Over time, b increases from zero to one or decreases from one to zero dependent on the required resolution. As a result, particles of one blend-domain smoothly replace the particles in their complementary blend-domain, their so-called **blend-partners**.

As described in Sec. 4.1, instead of just refining one global fluid volume, we blend between many **local blend-sets** (i.e. sub-volumes of the fluid) simultaneously as shown in Fig. 5. In each simulation step, particles first evaluate SPH quantities in their blend-domain only, as described in Sec. 4.2. Subsequently, blend-domains synchronize quantities by using an interpolation within their complementary blend-domain as described in Sec. 4.3. At the end of each simulation step, blend-weights are updated according to the estimated blending error as described in Sec. 4.4.



(a) (b)

Figure 5: *In 5(a) particles are split to 8 child particles in order to double resolution close to the pillars. With blend-sets, new (black) particles are smoothly blended in over time until they fully contribute to neighbouring particles (white) while contribution of the old particles is simultaneously reduced. In 5(b) particles are replaced by two child particles three times in order to double resolution. Due to blending and proper initialization the result is independent from the utilized pattern.*

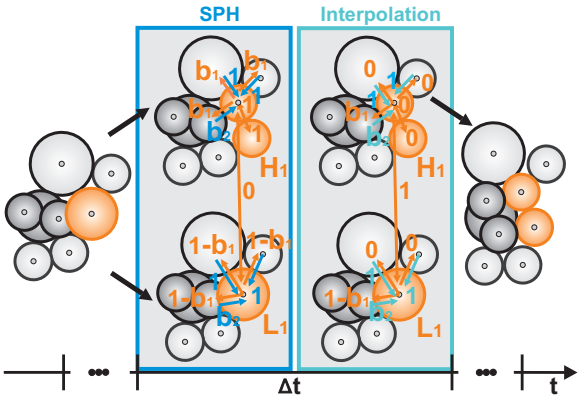


Figure 6: *During a simulation step Δt, a blending of particles requires two subsequent steps in order to update flow quantities: Particles blend between a quantity which they evaluate in their blend-domain (See Sec. 4.2) and a quantity which they interpolate within their complementary blend-domain (See Sec. 4.3). Both steps require different pair-wise contributions between neighbouring particles as shown for the orange blend-set.*

4.1. Blend-Sets

Blend-sets represent local fluid volumes with changing particle representations. As such, each blend-set s consists of two interchangeable particle sets, a low resolution particle set L_s and a high resolution particle set H_s, as shown in Fig. 5. The transition between these local blend-domains is controlled by a blend-weight $b_s \in [0,1]$ assigned to a blend-set.

Due to multiple co-existing blend-sets, the SPH system needs to handle several individually blending particles with changing contributions over time. As sketched in Fig. 6, two subsequent steps resemble Eq. (8) for blend-sets in order to account for such varying particle interactions: Firstly, particles compute a flow quantity $\widetilde{Q}_i$, like velocity or density, using SPH summations in their blend-domain as described in Sec. 4.2. Secondly, they interpolate quantities $\hat{Q}_i$ in their complementary blend-domain (See Sec. 4.3). Particles then blend both separately calculated quantities by using the blend-weight b_s of their blend-set s:

$$Q_i = \begin{cases} b_s\,\widetilde{Q}_i \;+\; (1-b_s)\,\hat{Q}_i & i \in H_s \\ (1-b_s)\,\widetilde{Q}_i \;+\; b_s\,\hat{Q}_i & i \in L_s \end{cases} \tag{9}$$

With such a blending of quantities, blend-partners synchronize flow dynamics between blend-domains which leads to a consistent and conservative transition over time. For example, in case of a split, $b_s = 0$ smoothly increases over time until $b_s = 1$ is reached. Consequently, newly created particles $i \in H_s$ gain influence on the local flow dynamics while simultaneously, old particles in L_s become passive over time. Vice versa, in case of a merge, the new particle in L_s consistently replaces the old particles in H_s. Please note that particles which are not part of a blend-set do not need any synchronization, thus, $Q_i = \widetilde{Q}_i$.

However, both operations, the quantity summation and the subsequent quantity interpolation step, imply different "visibilities" among neighbouring particles. Consequently, particles apply blend-weights with respect to the current operation and with respect to their blend-domain as described in the next sections.

4.2. Quantity Summation per Blend-Domain

In order not to change quantity fields abruptly, particles smoothly change their contribution to neighbouring particles by using the blend-weight b_s of their blend-set s. We enforce such a smooth transition by integrating pair-wise contributions $\widetilde{b}_{j\to i}$ into the SPH summation interpolants (Eq. 1), which results in:

$$\widetilde{Q}_i = \sum_j \widetilde{b}_{j\to i}\, Q_j\, \frac{m_j}{\rho_j}\, W_{ij}. \tag{10}$$

A contribution of a particle j onto it's neighbouring particle i is defined by:

$$\widetilde{b}_{j\to i} = \begin{cases} 0 & j \in H_s \wedge i \in L_s \;\vee\; j \in L_s \wedge i \in H_s \\ b_s & j \in H_s \wedge i \notin s \\ 1-b_s & j \in L_s \wedge i \notin s \\ 1 & \text{else.} \end{cases}$$

As blend-partners represent the same fluid volume, they do not contribute to each other during a SPH summation step as shown in Fig. 6. Instead, blend-domains exchange flow information afterwards by employing a quantity interpolation. In

contrast, particles of the same blend-domain fully contribute to each other during a SPH summation. Note that particles j, which are not part of a blend-set, fully contribute to all of their neighbouring particles, i.e. $b_{j\to i} = 1$.

With such pair-wise contributions $\widetilde{b}_{j\to i}$ the system smoothly avoids instantaneous changes in the density field which otherwise would lead to strong pressure forces:

$$\widetilde{\rho}_i = \sum_j \widetilde{b}_{j\to i}\, m_j\, W_{ij}.$$

As contributions are applied as an inherent particle property, they do not change the way spatial derivatives are computed. For example pressure forces (See Eq. (2)) are evaluated by:

$$\widetilde{\mathbf{F}}_i^p = -m_i \sum_{j\neq i} \widetilde{b}_{j\to i}\, m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}) \nabla W_{ij}\ .$$

As blend weights are particle inherent properties, they are easily apply to all other kinds of symmetrized gradient approximations. Consequently, the convective flux is computed by

$$\frac{\partial \widetilde{\mathbf{v}}_i}{\partial t} = \frac{1}{m_i}[\widetilde{\mathbf{F}}_i^p + \widetilde{\mathbf{F}}_i^\mu + \widetilde{\mathbf{F}}_i^\sigma + \widetilde{\mathbf{F}}_i^e].$$

Similarily, the diffusive flux (See Eq. (4)) with support for blend-sets is computed by:

$$\frac{\partial \widetilde{c}_i}{\partial t} = D \sum_{j\neq i} \widetilde{b}_{j\to i} \frac{m_j}{\rho_j \rho_i} (c_i - c_j) |\nabla W_{ij}|.$$

With the described SPH summations neighbouring particles smoothly adapt to a new particle configuration. However, particles within a blend-domain still receive a full contribution from their neighbouring particles which still might lead to pressure changes. We smooth these remaining sampling errors by utilizing a quantity interpolation between blend-domains.

4.3. Quantity Interpolation between Blend-Domains

In each step of the simulation, a blending of quantities between blend-domains of a blend-set reduces divergence among blend-domains and smooths sampling errors between blend-partners. However, particles cannot use the SPH summation interpolant directly in order to interpolate accurate quantities in the complementary blend-domain as Eq. (1) does not preserve constant functions and is sensitive to a varying number of neighbouring particles, e.g. at free surfaces. Instead, we utilize a constant correction as proposed in [BK02]In combination with pair-wise contributions particles then interpolate flow quantities $\widetilde{Q}_i$, like velocities $\widetilde{\mathbf{v}}_i$ or densities $\widetilde{\rho}_i$, in their complementary blend-domain by:

$$\hat{Q}_i = \frac{\sum_j \hat{b}_{j\to i}\, \widetilde{Q}_j\, \hat{W}_{ij}}{\sum_j \hat{b}_{j\to i}\, \hat{W}_{ij}} \tag{11}$$

During this step, the contributions from neighbouring particles are defined by:

$$\hat{b}_{j\to i} = \begin{cases} 0 & i,j \in L_s \ \vee \ i,j \in H_s \\ b_s & j \in H_s \wedge i \notin s \\ 1-b_s & j \in L_s \wedge i \notin s \\ 1 & \text{else.} \end{cases}$$

In contrast to the previous summation step, the interpolation is applied over blend-partners as well as neighbouring particles which either do not undergo any transition or belong to a different blend-set. In order to give particles some more space to adapt we slightly increase the interpolation radius:

$$\hat{W}_{ij} = W(|\mathbf{x}_i - \mathbf{x}_j|, k\,\frac{h_i + h_j}{2})$$

where $k = 1.25$ in all our examples. However, in rare cases, blend-partners may still diverge, e.g. due to contact with sharp boundaries or due to a strong divergence between neighbouring particles. In case blend-partners appear to loose contact, i.e $|\mathbf{x}_i - \mathbf{x}_j| > k\,\frac{h_i+h_j}{2}$, we average the velocity among blend-partners in order to let them stick together. On the one hand this effectively reduces their dynamics but on the other hand avoids non-physically motivated bonding mechanisms.

Please note that with interpolation we get a good approximation but do not directly preserve linear nor angular momentum.As a solution, one could employ a normalization of the interpolated quantities afterwards. As the introduced damping of flow dynamics is not noticeable we do not apply a subsequent normalization of the interpolated density and velocity values. However, we must conserve the total amount of a soluble substance c just as we conserve a fluid's mass. Fortunately, due to isotropic diffusion, the concentration profile is rather homogeneous. As concentrations are given with respect to a particles mass, we conserve concentrations by:

$$\hat{c}_i = \sum_j \widetilde{c}_j \tag{12}$$

where in this case the contributing particle set j is restricted to blend-partners only, i.e. $j \in H_s, i \in L_s$ or $j \in L_s, i \in H_s$.

4.4. Blending Duration

After each simulation step blend-sets smoothly update their blend-weights in order to enable a stable transition from one time-step to the next. Over time, the contribution of new particles increase from zero to one and simultaneously smoothly decrease from one to zero for old particles as modelled by the following piecewise linear blending function:

$$b_s(t+\Delta t) = b_s(t) + \begin{cases} \Delta b_s(t) & L_s \text{ splitted} \\ -\Delta b_s(t) & H_s \text{ merged} \end{cases} \tag{13}$$

where $\Delta b_s(t)$ is a error-related blend-increment of a blend-set. Note that re-sampling mechanisms, which have been utilized so far, correspond to a non-continuous replacement

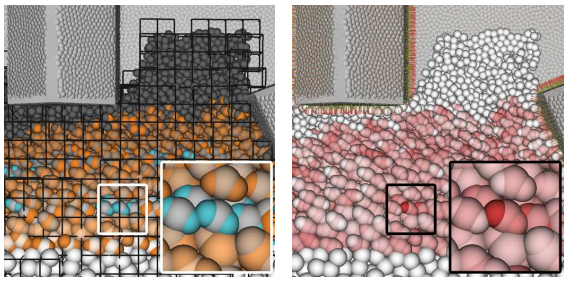


Figure 7: *Top-view (left) of the "Valley"-scene in combination with the estimated sampling-errors (right). Newly created (blue) particles may introduce large blend-errors (red), i.e. $E_{i,\rho}(\Delta b_{\max}) > E_{\max,\rho}$, due to strong overlaps with their neighbour particles.*

of particles, i.e. $\Delta b_s(t) = 1 \ \forall s,t$. However, we restrict blend-increments $\Delta b_s(t) \in [\Delta b_{\min}, \Delta b_{\max}]$, where $\Delta b_{\max} = \Delta t / T_{\min}$ and $\Delta b_{\min} = \Delta t / T_{\max}$, in order to give the user control over the blending duration with respect to the current integration time-step Δt as controlled by CFL conditions. By defining a minimum blending time $T_{\min}$ and a maximum blending time $T_{\max}$ a user can steer the blending either towards performance or towards accuracy. For all our examples we have set $T_{\min} = 40$ms and $T_{\max} = 200$ms which according to our experiments results in a smooth transition. As soon as old particles do not contribute to their neighbouring particles anymore, i.e. $b_s(t) = 1$ in case of a split or $b_s(t) = 0$ in case of a merge, they will be removed from the system.

In order to reduce sampling errors as shown in Fig.4, we increment blend-weights over time with respect to the introduced sampling error which we measure by applying a global blend-increment $\Delta b \in]0,1]$ for all blend-sets, which then results in the predicted contributions:

$$\widetilde{b}^*_{j\to i} = \begin{cases} 0 & j \in H_s \wedge i \in L_s \ \vee \ j \in L_s \wedge i \in H_s \\ b_s + \Delta b & j \in H_s \wedge \ i \notin s \\ 1 - (b_s + \Delta b) & j \in L_s \wedge \ i \notin s \\ 1 & \text{else.} \end{cases}$$

By assuming a static particle neighbourhood we are then able to measure the sensitivity of a quantity field with respect to a change of blend-weights:

$$\begin{aligned} E_{i,Q} = & \ | \widetilde{Q}_i - \widetilde{Q}^*_i | \\ = & \ | \textstyle\sum_j \widetilde{b}_{j\to i} Q_j \frac{m_j}{\rho_j} W_{ij} \ - \\ & \ \textstyle\sum_j \widetilde{b}^*_{j\to i} Q_j \frac{m_j}{\rho_j} W_{ij} \ | \end{aligned}$$

As particles which do not undergo blending naturally do not introduce sampling errors, we can reduce the summation to blending particles j only, which then simplifies the estimation of the sampling error to:

$$E_{i,Q} = \sum_j \Delta b \, Q_j \frac{m_j}{\rho_j} W_{ij}$$

Even if we can deduce sampling errors for all quantity fields, we only measure the sensitivity of the density field, as it is most important to stability, by:

$$E_{i,\rho}(\Delta b) = \sum_j \Delta b \, m_j \, W_{ij} \ . \tag{14}$$

In each time step, one could now iteratively adjust Δb in order to stay below a maximum user defined error $E_{\max,\widetilde{\rho}}$. By assuming a linear dependency between blend-weights and the sampling error, we instead apply only a single error estimation step with respect to the current maximum allowed blend-increment $\Delta b = \Delta_b \max$ (See Fig.7). After estimation of individual sampling errors a blend-set then computes its blend-increment by:

$$\Delta b(t) = \Delta b_{\max} - (\Delta b_{\max} - \Delta b_{\min}) \ \min_j \frac{E_{j,\rho}(\Delta b_{\max})}{E_{\max,\rho}} \tag{15}$$

where j includes blending particles as well as non-blending neighbouring particles of a blend-set and $E_{\max,\rho} = 0.06 * \rho_0$ in all our examples.

However, as shown in Fig.7, newly created particles may introduce larger errors than $E_{\max,\rho}$ due to excessive overlaps with neighbouring particles or due to particle boundaries. We improve their initial positions over 10 simulation steps by using their pressure force and keeping $b_i = 0$:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t^2 \frac{1}{m_i} \widetilde{\mathbf{F}}^p_i \, . \tag{16}$$

We restrict such virtual particles not to move more than h_i away from the mass center of their blend-partners. Please note that such impulses can only improve positions to a certain extent. A merge or a split is postponed if the error is still too high, i.e. $E_{i,\rho}(\Delta b_{\max}) > E_{\max,\rho}$.

5. Implementation

We have implemented the state-of-the-art PCISPH algorithm [SP09] in combination with a diffusive flux [CM99] on the GPU as shown in Alg. 1. Changes to the standard algorithms due to blend-sets are highlighted in orange. As can be seen, particles of a blend-set synchronize their density, velocity and concentrations among blend-domains as described throughout Sec. 4.1-4.3. Please note that we do not include a blending of forces. Instead, we synchronize the convective flux at the level of velocity. At the end of each simulation step blend-sets update their blend-weights by predicting a sampling error as described in Sec. 4.4.

In contrast to the diffusive flux, which is implemented in a straight-forward fashion, the convective flux has to iteratively solve an equality constraint $\rho_i(t + \Delta t) - \rho_0 = 0$ in order to enforce incompressibility. On the GPU, the termination of the loop could be evaluated by an parallel add-reduction over the particle's convergence decisions. However, in practice, three steps have turned out to be sufficient [IAGT10]. Since we utilize non-uniform particle sizes we pre-compute constants

Algorithm 1: Parallel-adaptive PCISPH simulation step.

Diffusive Flux

foreach *particle* i ***inparallel*** **do**
- compute density $\widetilde{\rho}_i(t)$

foreach *blended particle* i ***inparallel*** **do**
- interpolate density $\hat{\rho}_i(t)$(Eq. (11))
- synchronize density $\rho_i(t)$(Eq. (9))

foreach *particle* i ***inparallel*** **do**
- update concentration $\widetilde{c}_i(t+\Delta t)$

foreach *blended particle* i ***inparallel*** **do**
- interpolate concentration $\hat{c}_i(t+\Delta t)$(Eq. (12))
- synchronize concentration $c_i(t+\Delta t)$(Eq. (9))

Convective Flux

foreach *particle* i ***inparallel*** **do**
- compute non-pressure forces $\widetilde{\mathbf{F}}_i^{\mu+e+\sigma}(t)$
- initialize pressure $p_i(t)=0$
- initialize pressure force $\widetilde{\mathbf{F}}_i^{p}(t)=\mathbf{0}$

while $n<3$ **do**
- **foreach** *particle* i ***inparallel*** **do**
  - predict velocity $\mathbf{v}_i^*(t+\Delta t)$
  - predict position $\mathbf{x}_i^*(t+\Delta t)$
- **foreach** *particle* i ***inparallel*** **do**
  - predict density $\widetilde{\rho}_i^*(t+\Delta t)$
- **foreach** *blended particle* i ***inparallel*** **do**
  - interpolate density $\hat{\rho}_i^*(t+\Delta t)$(Eq. (11))
  - synchronize density $\rho_i^*(t+\Delta t)$(Eq. (9))
- **foreach** *particle* i ***inparallel*** **do**
  - compute pressure force $\widetilde{\mathbf{F}}_i^{p}(t)$
  - correct pressure $p_i(t)$ += $\kappa_l(\rho_i^*(t+\Delta t)-\rho_0)$

foreach *particle* i ***inparallel*** **do**
- update velocity $\widetilde{\mathbf{v}}_i(t+\Delta t)$

foreach *blended particle* i ***inparallel*** **do**
- interpolate velocity $\hat{\mathbf{v}}_i(t+\Delta t)$(Eq. (11))
- synchronize velocity $\mathbf{v}_i(t+\Delta t)$(Eq. (9))

Error Estimation

foreach *particle* i ***inparallel*** **do**
- estimate blend error $E_{i,\rho}(t)$(Eq. (14))

foreach *blend-set* s ***inparallel*** **do**
- compute blend increment $\Delta b_s(t)$(Eq.(15))
- update blend weights $b_s(t+\Delta t)$(Eq. (13))

foreach *blended particle* i ***inparallel*** **do**
- if $b_s(t)=0 \vee b_s(t)=1$: relax position $x_i(t)$(Eq. (16))

Time Integration

foreach *particle* i ***inparallel*** **do**
- update position $\mathbf{x}_i(t+\Delta t)$
- estimate time step Δt_i

adapt Δt using parallel Min-Reduction over Δt_i

spatial discretization directly implies adaptive temporal discretization, particles evaluate their own time-step restriction Δt_i by using Eq. (6). The overall integration time step Δt is then smoothly adapted using a parallel min-reduction over all individually evaluated time-step restrictions. For reasons of simplicity, we refer to the excellent related work for a description of the predictive-corrective convection loop [SP09] in combination with adaptive time-stepping and particle boundaries [IAGT10].

In order to improve memory coherence, particles are sorted according to their current indices into a regular hash grid, using a radix sort algorithm on the GPU [Gre09, Gra08]. Bins of the hash grid have a size equal to the maximum support radius $h_{\max}$, as visualized in Fig. 7. Sorting allows the system to easily insert or remove particles at the end of the respective linear data arrays, without taking care of memory fragmentations. Similar to Pelfrey et al. [PH10], we build neighbour lists for each particle including blend partners in order to speed-up SPH summations and interpolations. On the one hand such reference lists increase memory requirements and need to be updated each time particles have been sorted. But on the other hand, subsequent operations compute much faster as global memory reads for non-contributing particles are avoided. Furthermore, SPH operations are easier to implement, require less computing resources and avoid branch divergences on the GPU as they do not need an optimized neighbour search.

Scene	**"Valley"**	**"Pillars"**	**"Teapot"**
Sim. Time [s]	75	21	35
Avg.Δt [ms]	2	2.5	2
Min #ptcls [k]	0-500 / 0-1000	210,500,700 / 500,1000,1500	0-470 / 0-1000
	0-270*	380*,820*,1100*	0-700*
Comp. Time [min]	**34 / 57**	**5.9,13.7,18.1 / 7.6,17.3,26.7**	**15.8 / 19.8**
	32.3*	**9.9*,23.9*,31.3***	**18.68***
Snapshot	**Fig.1 at 30s**	**Fig.8 at 5s**	**Fig.3 at 4.5s**
#ptcls [k]	310[20]/ 635	500[40]/ 960	100[40]/ 128
Neighbours [ms]	10.1[0.7]/ 18.2	14.6[1.4]/ 26.5	5.2[2]/ 4.1
Diff. Flux [ms]	6.9[0.3]/ 12.4	9.67[1.1]/ 16.9	1.7[0.6]/ 2.1
Conv. Flux [ms]	27.1[3.3]/ 47.7	41.1[3.8]/ 65.7	11[2.5]/ 9.5
Est. Error [ms]	5.1[5.1]/ 0	9[9]/ 0	1.7[1.7]/ 0
Time Int. [ms]	1.1[0]/ 2.1	1.9[0]/ 4	1.9[0]/ 0
Split/Merge [ms]	0.7[0]/ 0	1.7[0]/ 0	0.2[0]/ 0
Total [ms]	**51[9.4]/ 80.4**	**78.1[15.3]/ 113.1**	**20.5[6.8]/ 16.6**

Table 1: *Timings of our adaptive / a non-adaptive PCISPH. The overhead due to blend-sets is highlighted in orange. Overall timings for [APKG07] are marked with *.*

$\kappa_l,\ l=0,1,..,l_{\max}$ to correct the density errors. Each is precomputed independently for a prototype particle of level l with a filled neighbourhood with only level-l particles. During pressure correction, each particle has to choose it's corresponding constant κ_l according to it's level l. Note that according to our measurements, it is sufficient to only synchronize a particle's density within the correction loop. As an adaptive

6. Results and Discussion

We have tested our scenes on an Intel Dual-Core 2.66 GHz with a NVidia GTX 580 Graphics Card with 1.5 GB VRAM. In order to demonstrate the applicability of our temporal blending we compare our method to the technique from Solenthaler et al. [SP09], Müller et al. [MCG03] and Adams et

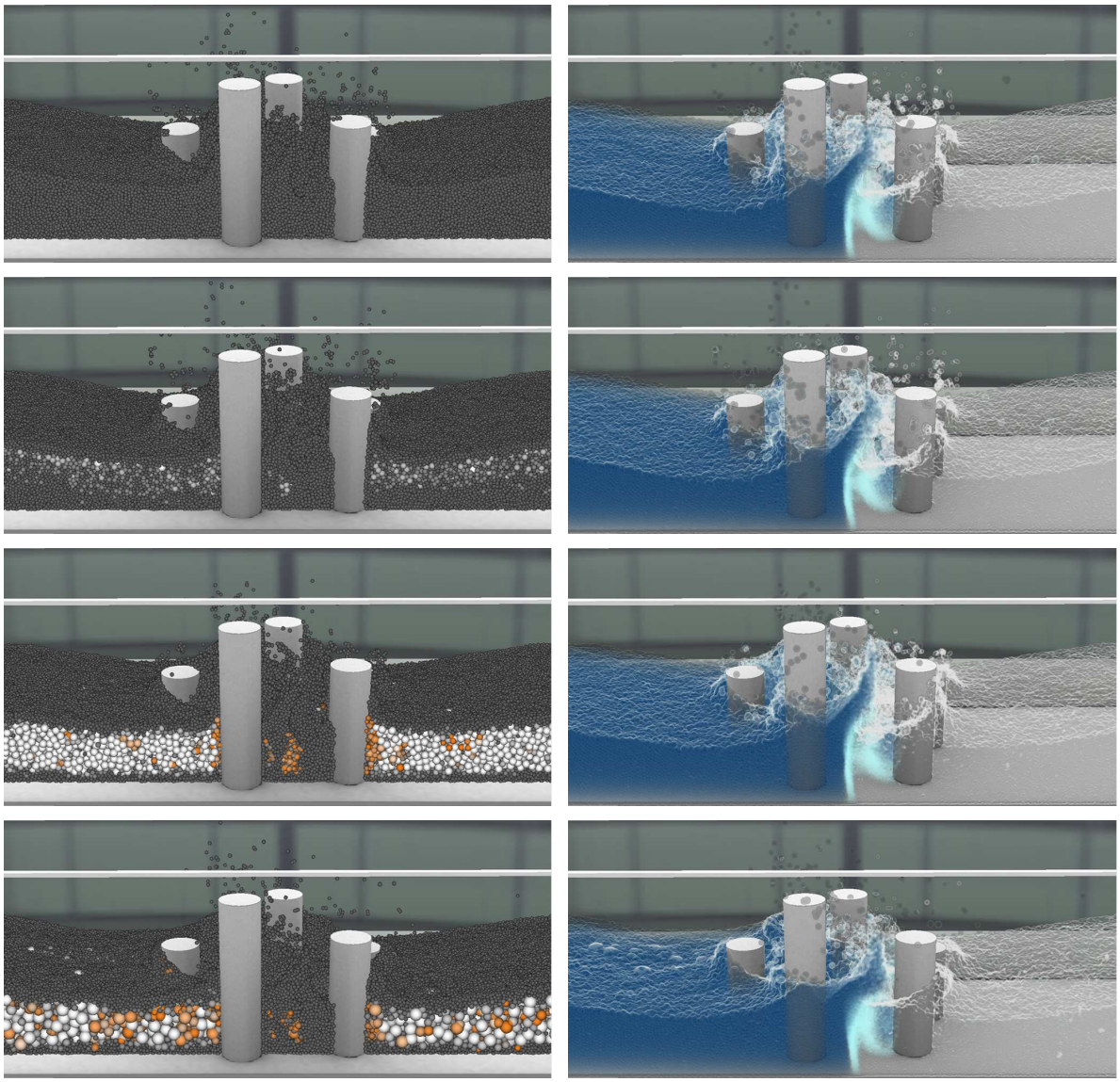


Figure 8: *PCISPH simulation of two colliding fluid fronts after 5 seconds (top row). Only 160k of the 960k particles can be stably merged with a non-continuous replacement of particles according to Adams et al. (second row) in combination with our relaxation (Eq.* (16)*). In contrast, our temporal blending (third row) achieves a speed-up of* 1.6 *by halving resolution with similar visual results. Increasing the particle resolution further to level* $l_{\max} = 6$ *(last row) significantly damps the flow dynamics.*

al. [APKG07] with particle numbers varying from $500k$ particles to $1,5M$ particles. Table 1 gives an overview over the simulation times for all presented scenes and shows timings for the operations as presented in Alg.1. Results are visualized by using an interactive volume ray-casting.

In the "Village" scene, Fig.1, the particle count increases to one million particles over time in case of a non-adaptive simulation and $500k$ in case of a comparable adaptive SPH within a fixed pre-defined high-resolution region around the village. As shown in Fig. 9, the workload scales linearly with the number of particles in the predictive-corrective loop. As expected, our adaptive method clearly outperforms the PCISPH [SP09] method by a factor of 1.6 and speeds-up a compressible SPH simulation [MCG03], by a factor of 1.4 when doubling resolution. As shown in Table1, the overhead due to blending is comparatively low and extra workload is only introduced by the error estimation step, as it is executed for all particles. Such workload could be avoided by just using a constant lin-

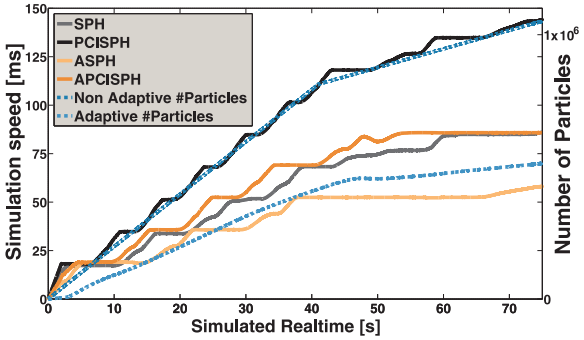


Figure 9: *Due to the linear dependency between performance and the number of particles our temporal blending ($\lambda_{\max} = 3$) speeds-up the simulation in Fig.1 by a factor 1.6 in case of PCISPH [SP09] and by a factor of 1.3 in case of SPH [MCG03].*

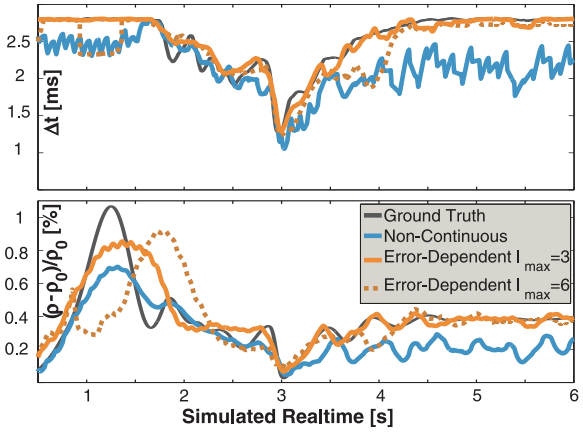


Figure 10: *Time-step size (top) and average density (bottom in percentage of the rest density) for methods presented in Fig. 8. Even if all adaptive methods preserve the density profile, a stable non-continuous replacement (blue) results in a flickering of the integration time-step (top) due to occasional high pressure forces. In contrast, our temporal blending still preserves the integration time-step, even for $l_{\max} = 6$.*

ear blending, but which result in either longer blending times or larger errors as shown in Fig. 4. With an error-estimation, the speed-up is still up to factor of 1.5 in case of dynamically changing high-resolution regions (which we recompute every 20 frames in order to effectively distribute the workload to the intermediate frames). However, in very complex flow scenarios the number of blend-sets might be very high compared to the overall particle count, as in the beginning of our Teapot-example (See Fig.3). As blend-sets need some time to adapt to a specific situation the overall speed-up is comparably low. Still, resolution regions are in good agreement with the flow dynamics.

By inserting particles abruptly [APKG07] high artificial pressure forces are introduced as visualized in Fig.4 which may result in an unstable simulation. In general, only very few particles can be stably re-sampled as shown in the second row of Fig. 8. Even if the average density error is less then 0.4% compared to a non-adaptive PCISPH, a non-continuous sampling still introduces high occasional pressure forces which results in a flickering of the integration time-step as shown in Fig. 10. In contrast, with our temporal blending the integration time-step is preserved while the number of particles can be halved as shown in the third row of Fig. 8. Unfortunately, in case of $l_{\max} = 6$ as shown in the last row of Fig.8, we cannot receive much speed-up compared to non-adaptive methods, since the cell-size which depends on the maximum support complicates neighbour search which then becomes a new bottleneck. Additionally, neighbour lists dictate the overall memory consumption. Please note that in our examples we reserve space for 40 neighbours per particles, including references for boundary particles as well. However, more sophisticated hash structures should solve the problem. Still, with an non-continuous replacement, it was not possible to stably merge particles up to level $l = 6$ even in case of highly viscous fluids. Beside refinement errors, they make a tracking of dynamic high resolution areas error-prone and that they damp flow dynamics notably as shown in the last row of Fig. 8. In the future we like to improve on the identification of high-resolution regions by addressing the error and including turbulent flow criterion. In it's current state, in our opinion, when doubling resolution, i.e. $l_{\max} = 3$, our adaptive simulation is in good agreement with a non-adaptive simulation.

7. Conclusion

We have presented a novel temporal blending approach which is capable of exchanging particle sets while maintaining a consistent convection-diffusion simulation while using standard SPH rules only. Our temporal blending is utilized for splitting and merging particles in order to adapt particle configurations according to flow requirements with minimal errors. We introduced a scheme to estimate the blending step size based on a predicted error in the pressure term in combination with an adaptive time-stepping. In order to evaluate the flexibility of our system we integrated a temporal blending in the latest approaches presented in the field of incompressible fluid simulations. We achieve interactive frame-rates with our fully GPU-based implementation for up to a million of coupled particles in combination with our blending approach.

References

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

[BK02] BONET J., KULASEGARAM S.: A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Appl. Math. Comput. 126*, 2-3 (Mar. 2002), 133–155.

[BOT01] BØRVE S., OMANG M., TRULSEN J.: Regularized smoothed particle hydrodynamics: A new approach to simulating magnetohydrodynamic shocks. *The Astrophysical Journal Supplement Series 561* (2001).

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 2008.

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA)* (2007), Eurographics Association, pp. 209–217.

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Trans. Vis. & Comput. Graph. 15*, 3 (2009), 493–503.

[CEL06] COLIN F., EGLI R., LIN F. Y.: Computing a null divergence velocity field using smoothed particle hydrodynamics. *J. Comput. Physics 217*, 2 (2006), 680–692.

[CKS00] COTTET G.-H., KOUMOUTSAKOS P., SALIHI M. L. O.: Vortex methods with spatially varying cores. *J. Comput. Phys. 162* (2000), 164–185.

[CM99] CLEARY P. W., MONAGHAN J. J.: Conduction modelling using smoothed particle hydrodynamics. *J. Comput. Physics 148* (1999), 227–264.

[CPK02] CHANIOTIS A. K., POULIKAKOS D., KOUMOUTSAKOS P.: Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *J. Comput. Physics 182* (2002), 67–90.

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (1996), Boulic R., Hegron G., (Eds.), Springer-Verlag, pp. 61–76.

[DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. Rep. 3829, INRIA, 1999.

[FB07] FELDMAN J., BONET J.: Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering 72*, 3 (2007), 295–324.

[GM77] GINGOLD R., MONAGHAN J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Notices of the Royal Astronomical Society 181* (1977), 375–389.

[Gra08] GRAND S. L.: *Broad-Phase Collision Detection with CUDA*, vol. GPU Gems 3. NVidia, 2008, ch. IV, pp. 697–721.

[Gre09] GREEN S.: *Particle Simulation using CUDA*. Tech. rep., NVIDIA, 2009.

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Eurographics Symp. on Computer Animation (SCA)* (2010), pp. 55–64.

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

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

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Proc. VRIPHYS* (2010), pp. 79–88.

[KAG*06] KEISER R., ADAMS B., GUIBAS L. J., DUTRÉ P., PAULY M.: *Multiresolution Particle-Based Fluids*. Tech. rep., ETH, 2006.

[KBKS09] KRISTOF P., BENES B., KRIVÁNEK J., STAVA O.: Hydraulic erosion using smoothed particle hydrodynamics. *Comput. Graph. Forum 28*, 2 (2009), 219–228.

[KC05] KOLB A., CUNTZ N.: Dynamic particle coupling for GPU-based fluid simulation. In *Proc. 18th Symposium on Simulation Technique,* (2005), pp. 722–727.

[KCR08] KOUMOUTSAKOS P., COTTET G.-H., ROSSINELLI D.: Flow simulations using particles: bridging computer graphics and cfd. In *ACM SIGGRAPH classes* (2008), pp. 25:1–25:73.

[KSW04] KIPFER P., SEGAL M., WESTERMANN R.: Uberflow: a GPU-based particle engine. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics Hardware* (2004), pp. 115–122.

[LB95] LAPENTA G., BRACKBILL J.: Control of the number of particles in fluid and mhd particle in cell methods. *Computer Physics Communications 87*, 1-2 (1995), 139 – 154.

[LQB05] LASTIWKA M., QUINLAN N. J., BASA M.: Adaptive particle distribution for smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids 46*, 10-11 (2005), 1403–1409.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics Sym. Computer Animation (SCA)* (2003), pp. 154–159.

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

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68* (2005), 1703–1759.

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 237–244.

[PH10] PELFREY B., HOUSE D.: Adaptive neighbor pairing for smoothed particle hydrodynamics. In *Proc. Int. Conf. Advances in Visual Computing (ISVC) - Part II* (2010), pp. 192–201.

[SAC*99] STORA D., AGLIATI P.-O., CANI M.-P., NEYRET F., GASCUEL J.-D.: Animating lava flows. In *Graphics Interface* (June 1999), pp. 203–210.

[SDG08] STANTCHEV G., DORLAND W., GUMEROV N. A.: Fast parallel particle-to-grid interpolation for plasma PIC simulations on the GPU. *J. Parallel Distrib. Comput 68*, 10 (2008), 1339–1349.

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulations. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30* (2011), 81:1–81:8.

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Trans. Graph. 28* (2009), 40:1–40:6.

[SZP07] SOLENTHALER B., ZHANG Y., PAJAROLA R.: Efficient refinement of dynamic point data. In *SPBG* (2007), pp. 65–72.

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Proceedings Symposium on Point-Based Graphics* (2008), pp. 137–146.