*Volume 31 (2012), Number 2 pp. 1–12* COMPUTER GRAPHICS *forum*

# Temporal Blending for Adaptive SPH

Jens Orthmann and Andreas Kolb

Computer Graphics Group, Institute for Vision and Graphics (IVG), University of Siegen, Germany

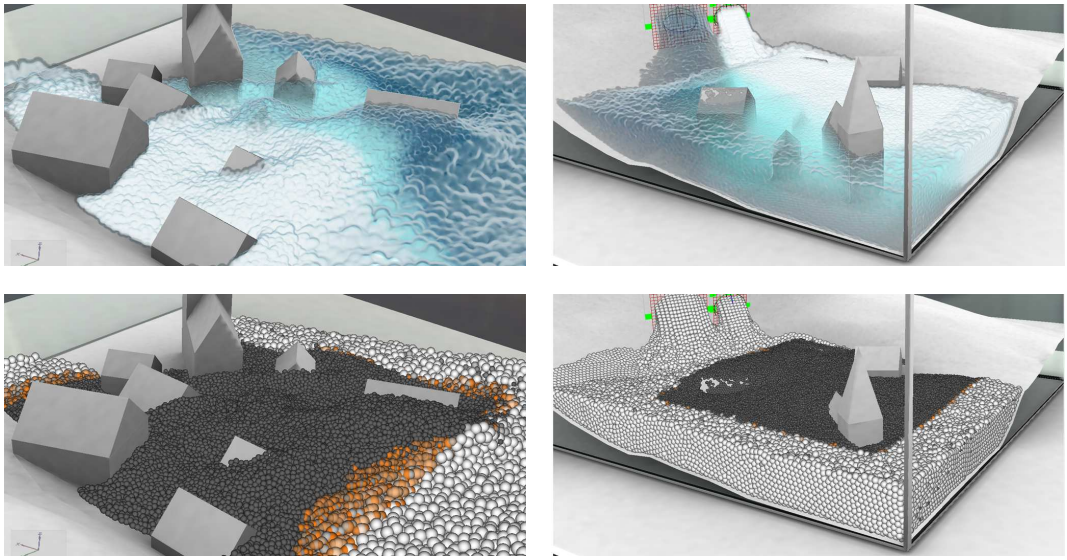


**Figure 1:** *Two fluids flooding a valley with a salt diffusion (from white to blue) in our interactive and fully GPU-based PCISPH, shown for two different time-steps. The particle resolution is smoothly doubled around the village by using blend-sets (orange).*

**Abstract**

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

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

## 1. Introduction

Fluid simulations are important in many graphics 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 and Teschner [BT07]. Different SPH parameters are crucial for the performance of an SPH simulation: 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 co-existing resolution levels [SG11]. Additionally, Predictive-Corrective Incompressible SPH (PCISPH) [SP09] enable large integration time-steps for incompressible fluids. Last but not least, graphics processing units (GPUs) are able to exploit the highly parallel nature of SPH simulations [KC05].

The overall goal of this paper is to provide a consistent and adaptive SPH simulation which allows large integration time-steps. For this purpose, we introduce a feature-specific adaptation of the number of particles in combination with adaptive time-steps and small support radii by using prediction-correction steps. An instantaneous replacement of particle configurations introduces a significant change in the pressure term, which leads to small time-steps when CFL conditions are dynamically enforced. Therefore, we apply a temporal blending scheme to smooth 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 enables 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 SPH fluids with support for non-uniform particle sizes and in combination with adaptive time-steps.

Compared to prior approaches, our temporal blending proves to be very robust in terms of the pressure error. As such, we believe that it is possible to integrate the proposed method into any other SPH scenario [LD09, SB12] where a smooth and consistent transition between particle sets is required.

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. 4-7. Implementation details are given in Sec. 8. In Sec. 9 we discuss advantages and limitations of our sampling and in Sec. 10 we conclude our approach.

## 2. Related Work

Since the introduction of SPH [GM77, Luc77], many improvements on computation speed have been introduced. We group them according to the complexity of the underlying physics, the data-parallelism, and the adaption of particle sets. For further details, we refer to the surveys from Monaghan [Mon05] and Koumoutsakos et al. [KCR08].

**Physics:** Desbrun and Cani [DC96] introduced SPH to the computer graphics community. Based on their work, Müller et al. [MCG03] presented a set of smoothing kernels with compact support in order to simulate fluids at interactive rates. Becker and Teschner [BT07] employed Tait's equation of state to enforce incompressibility at the cost of small time-steps. By introducing a prediction correction loop, Solenthaler and Pajarola [SP09] enabled larger integration time-steps for incompressible fluids. Ihmsen et al. [IAGT10] extended their predictive-corrective approach by globally adapting the integration time-step after each simulation step. Additionally, they have improved the boundary handling as proposed by Becker et al. [BTT09] by including boundary particles in the pressure update loop. Recently, Goswami and Pajarola [GP11] have introduced an approximation mechanism by deactivating nearly passive particles, however violating Newton's action-reaction principle. Beside convective flux, diffusion for SPH has been introduced to computer graphics by Stora et al. [SAC*99] to animate lava flows which then has been extended by Müller et al. [MSKG05] to simulate fluid-fluid interactions. 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 Neighbourhood Complexity:** Kolb and Cuntz [KC05] and Kipfer et al. [KSW04] were the first who took advantage of the data parallel nature of particle systems on programmable graphics hardware. The first gathering approach on the GPU has been published by Harada et al. [HKK07], incorporating a bucket structure on the GPU to accelerate neighbourhood searches. Green [Gre09] has taken this idea one step further by sorting particles according to their current location in a fixed size access-grid. Goswami et al. [GSSP10] further speed up their access structure by utilizing shared memory on the GPU and by applying z-indexing. Ihmsen et al. [IABT11] utilize a compact hashing on Multicore CPUs and make use of the temporal coherence of particle structures. Pelfrey and House [PH10], maintain neighbour lists 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 globally or locally is very appealing in order to improve simulation efficiency, because for SPH the overall computational cost increases linearly with the number of particles. Cot-

tet et al. [CKS00] divide the fluid domain into several areas with different resolutions. Unfortunately, their system depends on a frequent global remeshing of particles onto a regular grid [CPK02] which distracts from the adaptive character of Lagrangian systems. Lastiwka et al. [LQB05] relate a particle's support radius to the estimated volume after a local sampling using large support radii. Feldmann and Bonet [FB07] utilize a non-linear minimization of the local density error. However, both methods are too slow for interactive simulations. Instead, in computer graphics simple replacement schemes are used to enable fast simulations which cause high approximation errors. Desbrun and Cani [DC99] and Zhang et al. [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. With the differential form, errors are accumulated over time which leads to severe instabilities. As first published by Becker and Teschner [BT07], the summation approach is much more stable for larger time steps. Adams et al. [APKG07] determine valid positions for refined particles and stop sampling in case of large pressure errors. Keiser et al. [KAG*06] avoid a direct communication between particle levels by introducing the concept of virtual particles which however introduce errors to quantity fields when they turn into real particles. Recently, Solenthaler and Gross [SG11] have proposed a particle simulation with coexisting resolution levels which enables the simulation to quadruple resolution. Unfortunately, the proposed feedback forces are not physically motivated and do not prevent divergence between resolution levels. Divergent simulations 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 consistent multi-level convection-diffusion simulations.

## 3. Adaptive SPH

In this section, we first describe basics for SPH-based convection-diffusion simulations (Sec. 3.1) and then give insight into the identification of high resolution regions (Sec. 3.2) as required for adaptive SPH systems (Sec. 3.3), as shown in Fig. 2. In Sec. 3.4 we discuss major challenges in realizing an adaptive sampling while giving reasons for our proposed blending of quantities, as described in Sec.4.

### 3.1. SPH Physics

In Lagrangian systems, particles represent mass points which move with the flow field. In SPH, physical quantities are computed by summing up the contributions of particles $\mathbf{x}_j$ in the neighbourhood of the sampling position $\mathbf{x}$:

$$Q(\mathbf{x}) = \sum_j Q_j(\mathbf{x}) = \sum_j Q_j v_j W_j(\mathbf{x}), \tag{1}$$

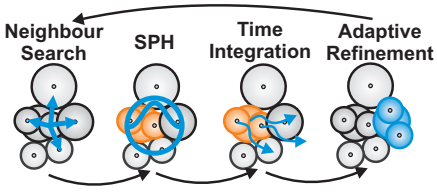



**Figure 2:** *A typical simulation loop for adaptive SPH systems. Our temporal blending easily integrates into SPH and uses an error estimation during the time-integration step in order to smooth out errors, caused by an adaptive sampling.*

where $v_j = \frac{m_j}{\rho_j}$ is a particle's volume, defined via a particle's mass $m_i$ and a particle's density $\rho_i$, and $W_j(\mathbf{x}) = W(|\mathbf{x}-\mathbf{x}_j|, h_j)$ is a radial symmetric kernel function with small compact support radius $h_j$ [MCG03]. If Eq. (1) is evaluated at a particle position $\mathbf{x}_i$ we will use the short notation $W_{ij} = W_j(\mathbf{x}_i)$. Consequently, in summation form, the density for a particle $i$ is computed by [Mon92]:

$$\rho_i = \sum_j m_j W_{ij}. \tag{2}$$

In our case, total mass transport of a soluble substance, like salt in Fig.1, is modelled via convective and diffusive transport. For fluids, the convective part or change of velocity 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 Monaghan [Mon05] and Becker and Teschner [BT07], respectively. However, a particle's pressure force $\mathbf{F}_i^p$ can be derived from symmetrized gradient approximations as described by Monaghan [Mon05]:

$$\mathbf{F}_i^p = -m_i \sum_{j \neq i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}, \tag{3}$$

where $p_i$ is the pressure of a particle which for PCISPH is adapted iteratively by utilizing the following linear dependency between density and pressure: $p_i = \beta(\rho_i - \rho_0)$, where $\rho_0$ is the rest density of a fluid and $\beta$ is precomputed over an optimal neighbourhood [SP09]. Additionally, an isotropic diffusion simulates how fast a soluble substance is propagated in a fluid. According to Cleary and Monaghan [CM99], a change of concentrations $c$ of a soluble substance is computed by

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

Here $D$ is the diffusivity constant, which controls how fast substances are propagated in a fluid. Please note that by separating $c$ from $m$, we assume that the soluble substance has no influence on a fluid's density.

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

### 3.2. High-Resolution Regions

An adaptive mechanism shifts computational resources to regions of interest. High-resolution regions are either predefined (see Fig. 1) or are changed dynamically, as dictated by the flow. Dynamic regions are defined via the fluid's surface in combination with areas of high diffusive flux, as visible at the contact-line between the two fluids shown in Fig. 3. Accordingly, particles $\mathbf{x}_i$ belong to a high-resolution area as long as one of the following conditions holds true:

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

where $\varepsilon_s, \varepsilon_c$ are user defined thresholds for the volume gradient (see also Mueller et al. [MSKG05]) and the concentration gradient, respectively. However, many other resampling criteria exist throughout the literature [SZP07]. Please note that similar to Adams et al. [APKG07] we leave an intermediate area of particles which are not refined in order to avoid split-merge fluctuations.

### 3.3. Non-Uniform Support Radii

Once high-resolution regions are identified, simple sampling operators, such as shown in Fig.6, are used to refine a local particle set, aiming for a good performance. An adaptive sampling uses particles with non-uniform support radii $h_i$ and masses $m_i$ depending on their current level $l_i = 0, 1, 2, ..., l_{\max}$:

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

where $m_0$ is the reference mass for level zero particles and $\sigma \approx 1.3$ in order to conserve a fluid's volume [Mon05]. However, in non-uniform particle systems, particles may either contribute to particles of their level only [KAG*06, SG11], or may exchange information with all neighbour particles directly [APKG07]. In the latter case, which we use due to its simplicity, the influence of neighbouring particles needs to be averaged in order to symmetrize contributions [Mon92]:

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

Alternative averaging operators [DC99] may be applied as well. However, high resolution regions require smaller integration time-steps in order to secure simulation stability. According to the Courant-Friedrich-Levy (CFL) condition, the maximum time-step for a single particle $i$ is defined by

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

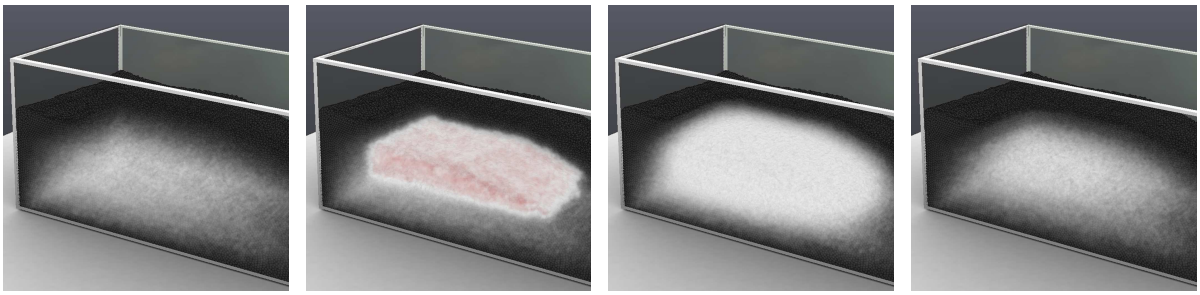
where $\lambda_v = 0.4$ and $\lambda_F = 0.25$ according to Monaghan [Mon92]. The overall simulation speed then depends on the minimum over all individual time-steps, as described for example by Ihmsen et al. [IAGT10] for PCISPH.

### 3.4. Challenges of an Adaptive Sampling

Adaptive SPH systems refine particles globally [CKS00] or locally [FB07]. In case of free surface flows, local sampling operations have to minimize a local error function in order to preserve a quantity field $Q(\mathbf{x})$ as good as possible:

$$E_Q(\mathbf{x}) = |Q(\mathbf{x}) - Q^*(\mathbf{x})|, \tag{7}$$

where $Q^*(\mathbf{x})$ is the result of the approximation. Lagrange multipliers [FB07] and iterative solvers are required in order to conserve the total amount of quantities and to account for non-negativity constraints [LB95]. Instead, in Computer Graphics, simple and fast sampling patterns are used. For example, in high-resolution regions, particles of level $l$ split to $N$ child particles of level $l - \log_2(N)$, or vice versa merge to a single particle of level $l + 1$ in low resolution regions. In general, such operators do not include any neighbouring particles to minimize $E_Q(\mathbf{x})$, do not preserve the overall regular particle structure and consequently, approximate the old quantity field with larger errors as shown in Fig. 4. Differentials in SPH are quite sensitive to such irregular particle structures and field discontinuities. Solenthaler and Gross [SG11] have utilized an impulse-based transition for high-level boundary particles turning into real particles. However, their method is applicable only if two resolution levels are allowed to coexist, possibly leading to divergence problems. In consistent systems, communication between particles of different smoothing radii increases the error as well [BOT01]. In general, this error can be reduced by avoiding a direct communication between levels as proposed by Keiser et al. [KAG*06] or by using a $1:2$ replacement structure as has been proposed by



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

**Figure 4:** *4(a) visualizes the pressure distribution for the scene shown in Fig.9 after 1.5 seconds of simulation time. In 4(b) high pressure is introduced due to an abrupt merging of 60k particles as utilized by [APKG07] in the context of PCISPH, leading to an unstable simulation in case the number of merge operations is not reduced as shown later in Fig.9. 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).*

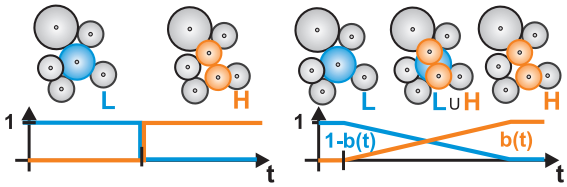
Adams et al. [APKG07]. Still, large pressure forces are introduced due to a non-optimized sampling, strong particle overlaps, and small compact smoothing kernels. By applying the CFL condition in each step, such forces dramatically decrease the integration time in order to preserve simulation stability. Even worse, in the context of PCISPH such forces may trigger a shock handling mechanism as described by Ihmsen et al. [IAGT10]. Instead, our temporal blending (Sec. 4) smooths out these errors over time by using an error-dependent transition (Sec. 7) via our local quantity blending (Sec. 5-6). Thus, we allow adaptive SPH systems to use large time steps in combination with small smoothing kernels which are required for a fast and consistent simulations.

## 4. Temporal Blending

In order to smooth sampling errors introduced by sampling operators, we propose a blending approach to smoothly interpolate between two interchangeable fluid representations over time as shown in Fig. 5. Even if a blending between multiple representations of an object is a well known concept in computer-graphics, we will transfer the idea into the context of SPH-based fluid simulation. With a blending between two global particle sets, the SPH-based summation interpolant changes to

$$\begin{aligned} Q(\mathbf{x}) =& b \sum_{j\in H} Q_j(\mathbf{x}) + (1-b) \sum_{j\in L} Q_j(\mathbf{x}) \\ =& b\, Q_H(\mathbf{x}) \quad + \quad (1-b)\, Q_L(\mathbf{x}). \end{aligned} \tag{8}$$

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 which depends 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**. Instead of just refining one global fluid volume, we blend between many **local blend-sets** (i.e. sub-volumes of the fluid) simultaneously as described in the following section.




(a) Non-Continuous (b) Continuous

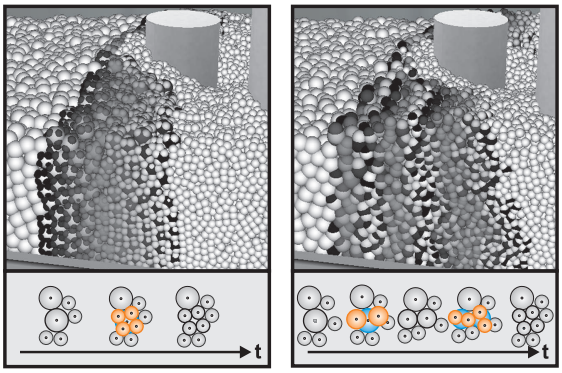
**Figure 5:** *Instead of a non-continuous sampling 5(a) our blending 5(b) smoothly replaces old particles (blue) by a new particle set (orange) with respect to a blend-weight b(t).*

## 5. Concept of Blend-Sets

We introduce the concept of blend-sets, i.e. local fluid volumes with adapting particle representations, to enable local blending operations. Each blend-set $s$ therefore consists of two particle sets, a low resolution particle set $L_s$ (blue) and a high resolution particle set $H_s$ (orange). The transition between these local particle-sets is controlled by a blend-weight $b_s \in [0,1]$. Due to multiple co-existing blend-sets, the SPH system needs to handle several individually blending particles together with non blending particles over time as shown in Fig. 6. Accordingly, the SPH summation with support for blend-sets is defined by

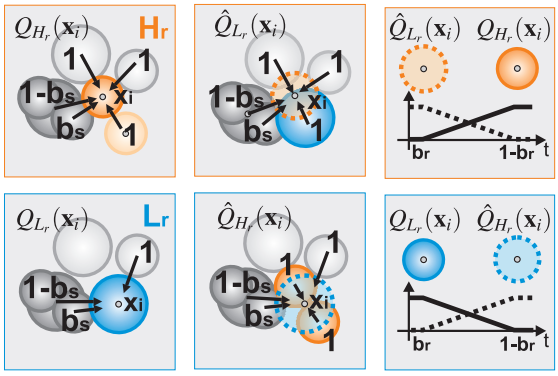
$$Q(\mathbf{x}) = \sum_{j\notin S} Q_j(\mathbf{x}) + \sum_s (b_s \sum_{j\in H_s} Q_j(\mathbf{x}) + (1-b_s) \sum_{j\in L_s} Q_j(\mathbf{x})), \tag{9}$$

where $S = \bigcup_s \{H_s \cup L_s\}$ includes all blending particles.




(a) (b)

**Figure 6:** *In 6(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 (white) to neighbouring particles while the contribution from old particles is reduced. In 6(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 used pattern.*




(a) SPH (b) Interpolation (c) Blending

**Figure 7:** *Particles i in a blend-set r blend (7(c)) between a quantity which they evaluate via SPH in their blend-domain (7(a)) and a quantity which they interpolate in their complementary blend-domain (7(b)), as shown for $i \in H_r$ (top row) and $i \in L_r$ (bottom row). In both steps, neighbouring blend-sets s (dark grey) contribute to a particle i with respect to their blend-weights $b_s$, as indicated by the black arrows.*

However, similarly to the global blending function as defined by Eq. (8), we need to compute flow quantities separately in both blend-domains of a blend-set. Consequently, we have to rearrange Eq. (9) with respect to a single blend-set $r$. Therefore, we define the contribution from particles which do not belong to $r$ (grey particles in Fig. 7) as

$$\Phi_r(\mathbf{x}) = \sum_{j \notin S} Q_j(\mathbf{x}) + \sum_{s \neq r} (b_s \sum_{j \in H_s} Q_j(\mathbf{x}) + (1-b_s) \sum_{j \in L_s} Q_j(\mathbf{x})) .$$

By adding this contribution to the contribution from particles in $r$ we can reformulate the SPH summation to

$$\begin{aligned} Q(\mathbf{x}) =& \Phi_r(\mathbf{x}) + b_r \sum_{j \in H_r} Q_j(\mathbf{x}) + (1-b_r) \sum_{j \in L_r} Q_j(\mathbf{x}) \\ =& b_r \left(\Phi_r(\mathbf{x}) + \sum_{j \in H_r} Q_j(\mathbf{x})\right) + (1-b_r) \left(\Phi_r(\mathbf{x}) + \sum_{j \in L_r} Q_j(\mathbf{x})\right) \\ =& b_r\, Q_{H_r}(\mathbf{x}) + (1-b_r)\, Q_{L_r}(\mathbf{x}), \end{aligned} \tag{10}$$

where $Q_{H_r}(\mathbf{x})$ and $Q_{L_r}(\mathbf{x})$ represent the quantity fields of the high-resolution blend-domain and low-resolution blend-domain with respect to a single blend-set $r$. However, in practice we cannot directly use Eq. (10) to compute mass flux, for reasons described in the following section.

## 6. Application of Blend-Sets

The separation of resolution levels as described in the previous section becomes useful when evaluating flow quantities for blend-sets. However, because blend-partners strongly overlap, a force computation in a particle's complementary blend-domain would lead to strong repulsion forces. That is why in practice, a particle $i$ in a blend-set $r$ utilizes three sequential steps to resemble Eq. (10), as shown in Fig. 7:

a) **SPH**: At first, particle $i$ computes its flow quantities via SPH in its blend-domain only, i.e. $Q_i = Q_{H_r}(\mathbf{x}_i)$ for $i \in H_r$ and $Q_i = Q_{L_r}(\mathbf{x}_i)$ for $i \in L_r$. During this step, blend-domains are treated independently (see Sec. 6.1).

b) **Interpolation**: Subsequently, particle $i$ interpolates flow quantities in its complementary blend-domain (see Sec. 6.2), resulting in $\hat{Q}_i = \hat{Q}_{H_r}(\mathbf{x}_i)$ for $i \in L_r$ or in $\hat{Q}_i = \hat{Q}_{L_r}(\mathbf{x}_i)$ for $i \in H_r$. However, an SPH summation would result in an underestimation of flow fields as the interpolation point $\mathbf{x}_i$ does not coincide with any particle position $\mathbf{x}_j$ in $i$'s complementary blend-domain. Instead, we apply a corrected interpolation [BK02]:

$$\hat{Q}(\mathbf{x}) = \sum_j Q_j v_j \hat{W}_j(\mathbf{x}), \quad \hat{W}_j(\mathbf{x}) = \frac{W_j(\mathbf{x})}{\sum_j v_j W_j(\mathbf{x})}, \tag{11}$$

which results in a better approximation of flow fields.

c) **Blending**: Finally, both quantities are blended with respect to the blend-weights $b_r$ in order to synchronize flow dynamics between blend-domains:

$$Q_i \leftarrow \begin{cases} b_r\, Q_i + (1-b_r)\, \hat{Q}_i & i \in H_r \\ b_r\, \hat{Q}_i + (1-b_r)\, Q_i & i \in L_r. \end{cases} \tag{12}$$

With such a blending of quantities, the system enables a smooth transition between blend-partners over time.

Instead of grouping particles according to their blend-sets we rather employ pair-wise conditions in order to evaluate flow quantities.

### 6.1. SPH-based Flux Computation per Blend-Domain

As a first step, a particle $i$ in blend-set $r$ evaluates flow quantities in its blend-domain. According to Eq. (10) neighbouring particles $j$, which belong to the same blend-domain, contribute to $i$ with respect to their blend-weights. Instead of gathering contributions from all blend-sets separately, we introduce conditional pair-wise contributions $b_{i\leftarrow j}$ into the SPH summation:

$$Q_i = \sum_j b_{i\leftarrow j}\, Q_j\, v_j\, W_{ij}. \tag{13}$$

As shown in Fig. 7(a), a contribution from a neighbouring particle $j$ to particle $i$ under consideration is then defined as

$$b_{i\leftarrow j} = \begin{cases} 0 & j \in H_s \wedge i \in L_s \;\vee\; j \in L_s \wedge i \in H_s \\ b_s & j \in H_s \wedge r \neq s \\ 1-b_s & j \in L_s \wedge r \neq s \\ 1 & \text{otherwise.} \end{cases}$$

Please note that pair-wise contributions $b_{i\leftarrow j}$ are also applied during flux computation for particles $i$ which do not belong to any blend-set. For such non-blending particles we simply set $r \neq s$, as $H_r \cup L_r = \emptyset$.

With such pair-wise contributions the SPH system avoids instantaneous changes in the density field (see Eq. (2)) which otherwise would lead to strong pressure forces:

$$\rho_i = \sum_j b_{i\leftarrow j}\, m_j\, W_{ij}. \tag{14}$$

Such conditional blend-weights do not change the way spatial derivatives are computed. For example pressure forces (see Eq. (3)) are evaluated by

$$\mathbf{F}_i^p = -m_i \sum_{j\neq i} b_{i\leftarrow j}\, m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}\ . \tag{15}$$

As blend weights are employed as particle inherent properties, they easily apply to all other kinds of symmetrized gradient approximations as well. Similarily, the diffusive flux (see Eq. (4)) with support for blend-sets is computed by

$$\frac{\partial}{\partial t} c_i = D \sum_{j\neq i} b_{i\leftarrow j} \frac{m_j}{\rho_j \rho_i} (c_i - c_j) |\nabla W_{ij}|. \tag{16}$$

With the described SPH summation, neighbouring particles smoothly adapt to new particle configurations. However, blend-partners need to synchronize their quantities by utilizing an interpolation in their complementary blend-domain.

### 6.2. Interpolation of Flow Quantities between Blend-Domains

In each step of the simulation, a blending of quantities reduces the divergence between blend-domains and smooths sampling errors between blend-partners. As described previously, particles utilize Eq. (11) in order to interpolate quantities in the complementary blend-domain as standard SPH does not even preserve constant functions. Consequently, in combination with pair-wise contributions, a particle $i$ in a blend-set $r$ interpolates flow quantities by

$$\hat{Q}_i = \frac{\sum_j \hat{b}_{i\leftarrow j}\, Q_j\, v_j\, W_{ij}}{\sum_j \hat{b}_{i\leftarrow j}\, v_j\, W_{ij}}. \tag{17}$$

As shown in Fig. 7(b), the contributions from neighbouring particles $j$ to particle $i$ under consideration is defined as

$$\hat{b}_{i\leftarrow j} = \begin{cases} 0 & i,j \in L_s \;\vee\; i,j \in H_s \\ b_s & j \in H_s \wedge r \neq s \\ 1-b_s & j \in L_s \wedge r \neq s \\ 1 & \text{otherwise.} \end{cases}$$

Please note that Eq. (17) is used for pure interpolation only, e.g. to interpolate velocities $\hat{\mathbf{v}}_i$ or densities $\hat{\rho}_i$. We also slightly increase the interpolation radius by $k = 1.25$ in order to get a good trade-off between the smoothing of quantity fields and the divergence between blend-partners:

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

However, in rare cases, blend-partners may still diverge, e.g. due to contact with sharp boundaries. In cases where $|\mathbf{x}_i - \mathbf{x}_j| > k\,\frac{h_i+h_j}{2}$, 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, an average operation avoids non-physically motivated bonding mechanisms.

Even if Eq. (11) results in a better approximation of quantity fields [She68] we neither preserve linear nor preserve angular momentum. As the introduced damping of flow dynamics is not noticeable we do not apply a normalization of the interpolated density and velocity values. However, we must conserve the total amount of a soluble substance $c$. Fortunately, due to isotropic diffusion, the concentration profile is rather homogeneous. As concentrations are given with respect to a particle's mass, we interpolate concentrations by

$$\hat{c}_i = \sum_j c_j, \tag{18}$$

where in this case the contributing particle set $j$ is restricted to blend-partners only, i.e. $j \in H_s \wedge i \in L_s \vee j \in L_s \wedge i \in H_s$. With the proposed blending of quantities, the system is able to smoothly exchange particle sets over time.

## 7. Blending Duration

After a blend-set has been created via a split or a merge operation, it goes through two different phases: in the initialization phase, newly created particles are passively advected while their initial position is improved, as described in Sec.7.2. In the subsequent transition phase, blend-sets smoothly update their blend-weights in order to enable a stable transition from one time-step to the next. The influence

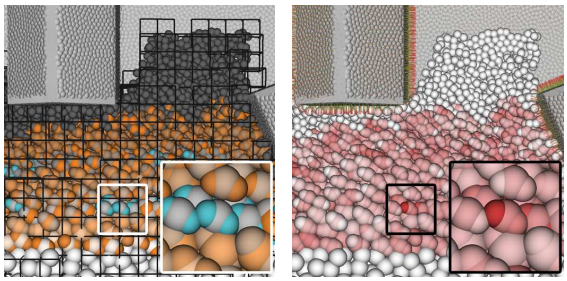


**Figure 8:** *Top-view (left) of the "Valley"-scene and the estimated sampling-errors (right). Newly created (blue) particles may introduce large blend-errors (red), due to strong overlaps with neighbouring particles.*

of new particles increases from zero to one, and simultaneously, decreases from one to zero for old particles as modelled by the following piecewise linear blending function:

$$b_r(t+\Delta t) = b_r(t) + \begin{cases} \Delta b_r(t) & L_r \text{ splitted} \\ -\Delta b_r(t) & H_r \text{ merged,} \end{cases} \tag{19}$$

where $\Delta b_r$ depends on the local sampling error (see Sec.7.1). As soon as old particles do not contribute to their neighbouring particles anymore, i.e. $b_r = 1$ in case of a split or $b_r = 0$ in case of a merge, they are removed from the system.

### 7.1. Error Estimation

In order to make the transition as smooth as possible, we increment blend-weights with respect to local sampling errors. For this purpose, we measure the error which is introduced into a quantity field by updating all blend-weights by a global blend-increment $\Delta b \in ]0,1]$, which according to Eq. (7) and Eq. (9) yields

$$\begin{aligned} E_Q(\mathbf{x}) =& \mid Q(\mathbf{x}) - Q^*(\mathbf{x}) \mid \\ =& \sum_s (b_s \sum_{j \in H_s} Q_j(\mathbf{x}) + (1-b_s) \sum_{j \in L_s} Q_j(\mathbf{x})) \\ & - \sum_s ((b_s+\Delta b) \sum_{j \in H_s} Q_j(\mathbf{x}) + (1-(b_s+\Delta b)) \sum_{j \in L_s} Q_j(\mathbf{x})) \\ =& \sum_s (\Delta b \sum_{j \in H_s} Q_j(\mathbf{x}) + \Delta b \sum_{j \in L_s} Q_j(\mathbf{x})) \\ =& \Delta b \sum_{j \in S} Q_j(\mathbf{x}) \ . \end{aligned}$$

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. Even if we can compute sampling errors for all quantity fields, we only measure the error which is introduced into the density field by

$$E_\rho(\mathbf{x}) = \Delta b \sum_{j \in S} m_j W_j(\mathbf{x}), \tag{20}$$

which in our examples is most important to ensure stability.

In each time step, one could iteratively adjust $\Delta b$ in order to stay below a maximum user defined error $E_{\rho,\max}$. However, by assuming a linear dependency between blend-weights and the sampling error, we instead apply only a single estimation step. At first, all particles evaluate the density error at their locations $\mathbf{x}_j$, resulting in $E_{\rho,j}$ (see Fig.8). During this step we set $\Delta b = \Delta b_{\max} = \Delta t / T_{\min}$ where the minimum blending time $T_{\min}$ is defined by the user. Afterwards, a blend-set computes its weight increment by using the maximum of all individual errors:

$$\Delta b_r = \Delta b_{\max} - (\Delta b_{\max} - \Delta b_{\min}) \max_j \frac{E_{\rho,j}}{E_{\rho,\max}}, \tag{21}$$

where $\Delta b_{\min} = \Delta t / T_{\max}$ and the maximum user defined density error $E_{\rho,\max} = 0.06 * \rho_0$. Please note that with particles $j$ we include non-blending neighbouring particles of blend-set $r$ as well. Both $T_{\max}$ and $T_{\min}$ allow a user to steer the blending either towards performance or towards accuracy. For all our examples we have set the minimum blending time $T_{\min} = 40$ms and the maximum blending time $T_{\max} = 200$ms, which according to our experiments results in a smooth transition. However, the blending duration can be greatly reduced if particles are initialized properly.

### 7.2. Particle Initialization

As shown in Fig.8, newly created particles may introduce larger errors than $E_{\rho,\max}$ due to excessive overlaps with neighbouring particles or due to a collision with rigid objects. Accordingly, we improve initial positions of new particles over a few simulation steps by using their pressure force and keeping $\Delta b_r = 0$:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \alpha \frac{1}{m_i} \mathbf{F}_i^p, \tag{22}$$

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

## 8. PCISPH Implementation with Blend-Sets

In order to validate our approach, we have implemented the state-of-the-art PCISPH algorithm in combination with a diffusive flux [CM99] on the GPU as shown in Alg. 1. For reasons of simplicity, we refer to the good related work for a description of the predictive-corrective convection loop [SP09] in combination with an adaptive time-stepping and proper boundary handling [IAGT10]. We have highlighted our modifications to introduce blend-sets (orange) and changes due to non-uniform support radii (green).

In each simulation step, particles blend their density, velocity and concentrations between blend-domains as described in Sec. 6. Please note that we do not blend forces

**Algorithm 1**: Parallel PCISPH step with blend-sets.

**Diffusive Flux**

**foreach** *particle i* ***in parallel*** **do**
  compute density $\rho_i(t)$ (Eq. (14))

**foreach** *blending particle i* ***in parallel*** **do**
  interpolate density $\hat{\rho}_i(t)$ (Eq. (17))
  blend density $\rho_i(t)$ (Eq. (12))

**foreach** *particle i* ***in parallel*** **do**
  update concentration $c_i(t+\Delta t)$ (Eq. (16))

**foreach** *blending particle i* ***in parallel*** **do**
  interpolate concentration $\hat{c}_i(t+\Delta t)$ (Eq. (18))
  blend concentration $c_i(t+\Delta t)$ (Eq. (12))

**Convective Flux**

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

**while** $n<3$ **do**
  **foreach** *particle i* ***in parallel*** **do**
    predict velocity $\mathbf{v}_i^*(t+\Delta t)$
    predict position $\mathbf{x}_i^*(t+\Delta t)$
  **foreach** *particle i* ***in parallel*** **do**
    predict density $\rho_i^*(t+\Delta t)$ (Eq. (14))
  **foreach** *blending particle i* ***in parallel*** **do**
    interpolate density $\hat{\rho}_i^*(t+\Delta t)$ (Eq. (17))
    blend density $\rho_i^*(t+\Delta t)$ (Eq. (12))
  **foreach** *particle i* ***in parallel*** **do**
    compute pressure force $\mathbf{F}_i^p(t)$ (Eq. (15))
    correct pressure $p_i(t)$ += $\beta_l(\rho_i^*(t+\Delta t)-\rho_0)$

**foreach** *particle i* ***in parallel*** **do**
  update velocity $\mathbf{v}_i(t+\Delta t)$

**foreach** *blending particle i* ***in parallel*** **do**
  interpolate velocity $\hat{\mathbf{v}}_i(t+\Delta t)$ (Eq. (17))
  blend velocity $\mathbf{v}_i(t+\Delta t)$ (Eq. (12))

**Blend-Set Transition**

**foreach** *particle i* ***in parallel*** **do**
  estimate blend error $E_{\rho,i}(t)$ (Eq. (20))

**foreach** *blend-set r* ***in parallel*** **do**
  compute blend increment $\Delta b_r(t)$ (Eq.(21))
  update blend-weights $b_r(t+\Delta t)$ (Eq. (19))

**foreach** *blending particle i* ***in parallel*** **do**
  if $b_r(t)=0 \vee b_r(t)=1$ : improve position $x_i(t)$ (Eq. (22))

**Time Integration**

**foreach** *particle i* ***in parallel*** **do**
  update position $\mathbf{x}_i(t+\Delta t)$
  estimate time step $\Delta t_i$
adapt $\Delta t$ using parallel Min-Reduction over $\Delta t_i$
recompute $\beta_l$

between blend-partners. Instead, we synchronize the convective flux by a blending of velocity values. At the end of each simulation step, blend-sets then update their blend-weights by predicting a sampling error as described in Sec. 7. In contrast to the diffusive flux, which is implemented in a straightforward fashion, the convective flux is solved by adapting particle pressures [SP09] over three iteration steps [IAGT10] in order to enforce incompressibility. Note that according to our measurements, it is sufficient to apply a synchronization of particle densities within the correction loop.

Since we utilize non-uniform particle sizes we compute multiple constants $\beta_l$, $l=0,1,..,l_{\max}$ to correct density errors. Similar to Solenthaler and Pajarola, each is precomputed independently for a prototype particle of level $l$ with a filled neighbourhood of level-$l$ particles and is updated each time the integration step-size changes. During iteration, each particle then chooses its constant $\beta_l$ corresponding to its level $l_i$. As an adaptive spatial discretization directly implies adaptive temporal discretization, particles evaluate their maximum time-step $\Delta t_i$ by using Eq. (6). The integration time step $\Delta t$ is then adapted globally by using a parallel min-reduction over all individual time-steps.

Before flux computation, particles are sorted according to their current location in a regular access grid [Gre09], whose size depends on the maximum support radius $h_{\max}$ (see Fig. 8). A sorting of particle data improves memory coherence and allows the system to easily insert or remove particles at the end of the respective linear data arrays which avoids memory fragmentations. Similar to Pelfrey and House [PH10], we then use the access grid to set up neighbour lists in which each particle stores references to all of its neighbouring particles. We identify particles as neighbours if their distance is smaller than their average support radius which results in a symmetric visibility during flux computations. On the one hand, such neighbour references increase memory requirements and need to be updated each frame, but on the other hand, SPH operations compute much faster as global memory reads for non-contributing particles are avoided. Furthermore, GPU kernels are easier to implement, require less computing resources and avoid branch divergences as they do not need an optimized neighbour search.

| **Scene** | **"Valley"** | **"Pillars"** | **"Teapot"** |
|---|---|---|---|
| Sim. Time [s] | 75 | 21 | 35 |
| Avg.$\Delta t$ [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.9 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 |
| Blend. Trans. [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 / non-adaptive PCISPH. The overhead due to blend-sets is highlighted in orange. Overall timings for [APKG07] are marked with *.*

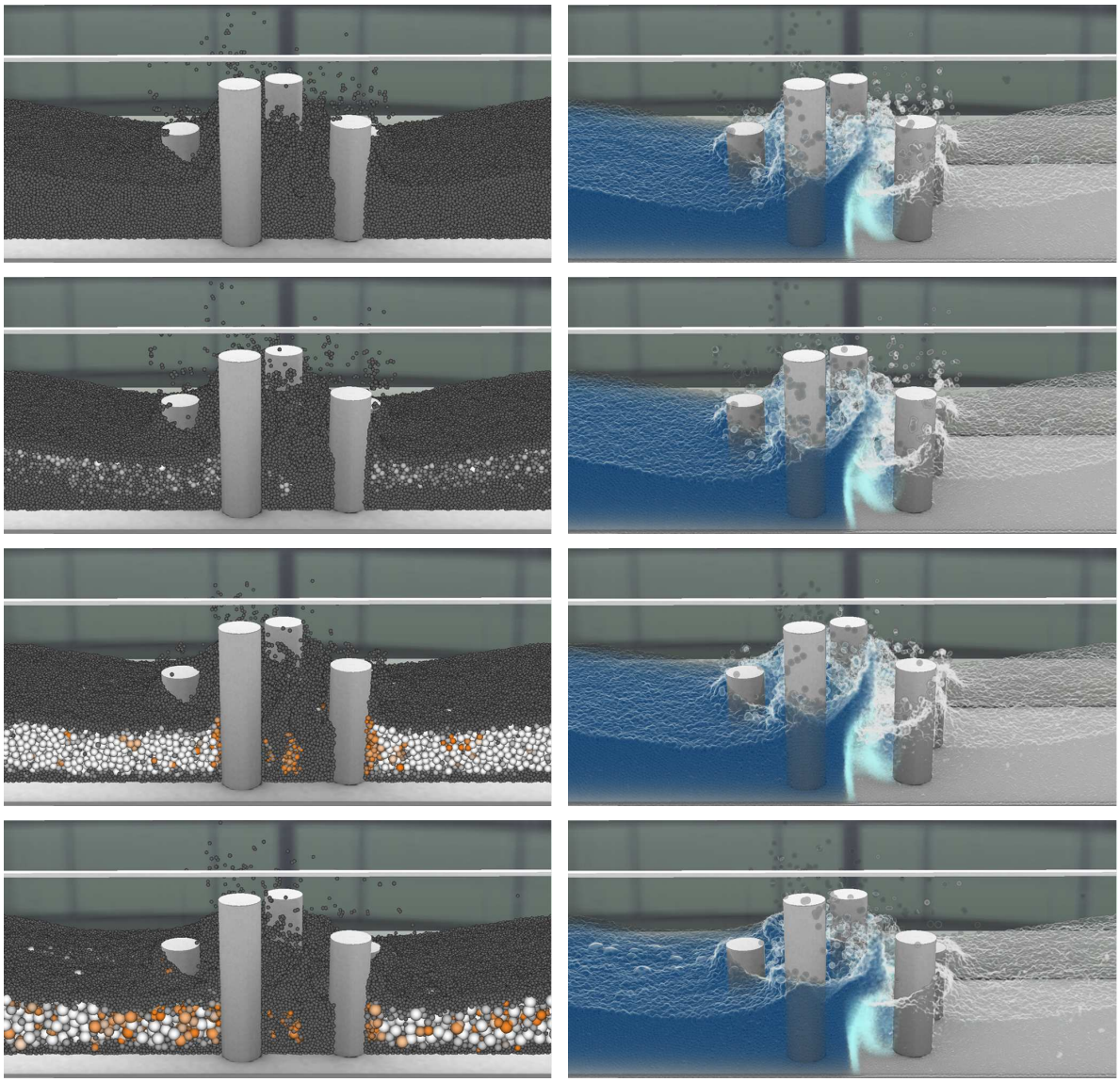


**Figure 9:** *PCISPH simulation of two colliding fluid fronts after 5 seconds (top row). Only 160 k of the 960 k particles can be stably merged with [APKG07] (second row). In contrast, our temporal blending (third row) achieves a 1.6x speed-up by halving the particle count. Decreasing the particle resolution further to level* $l_{\max} = 6$ *(last row) significantly damps flow dynamics.*

## 9. 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 techniques from Solenthaler and Pajarola [SP09], Müller et al. [MCG03] and Adams et al. [APKG07] with particle numbers varying from 500 k particles to 1,5 M 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 our interactive volume ray-casting [OKK10].

In the "Village" scene, Fig.1, the particle count increases to one million particles over time in case of a non-adaptive simulation and 500 k in case of a comparable adaptive SPH within a fixed predefined high-resolution region around the village. As shown in Fig. 10, the workload scales linearly with the number of particles in the predictive-corrective loop. As expected, our adaptive method outperforms the PCISPH [SP09] method by a factor of 1.6 and speeds-up

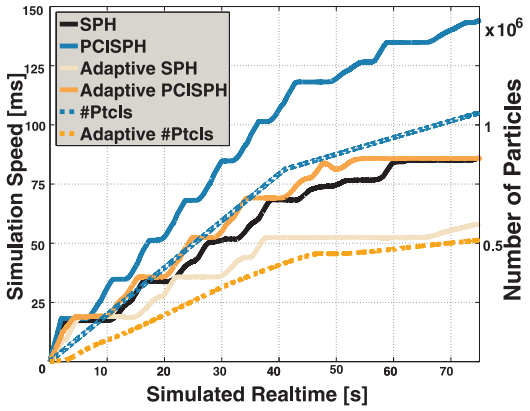



**Figure 10:** *Due to the linear dependency between performance and the number of particles, our temporal blending speeds-up the simulation in Fig.1 by a factor 1.6 for PCISPH [SP09] and by a factor of 1.3 for SPH [MCG03].*

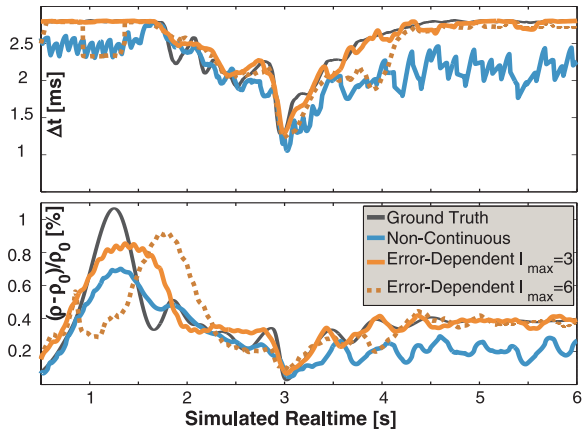



**Figure 11:** *Time-step size and average density (in percentage of the rest density) for Fig. 9. Even if the density profile is preserved a stable non-continuous replacement (blue) results in a flickering of the integration time-step (top) due to occasional high pressure forces. In contrast, our method still preserves the integration time-step, even for* $l_{\max} = 6$.

a compressible SPH simulation [MCG03], by a factor of 1.4 when the particle count is halved. As shown in the Teapot-example (see Fig.3), in very complex flow scenarios, the number of blend-sets might be very high compared to the overall particle count which results in no speed-up at all. However, this will be the case for most adaptive simulations due to fast changing resolution regions (which we recompute every 20 frames). Still, resolution regions are in good agreement with flow dynamics.

By inserting particles abruptly [APKG07], high artificial pressure forces are introduced, as visualized in Fig.4. In general, very few particles can be stably merged, as shown in Fig. 9. Even with a reduced number of refinement operations, occasional high pressure forces occure and lead to a flickering of the integration time-step, as shown in Fig. 11. The system is not able to stably merge particles up to level $l = 6$. In contrast, our temporal blending preserves the integration time-step while the number of particles can be halved, even for $l_{\max} = 6$ as shown in the last row of Fig.9. Unfortunately, in case of $l_{\max} = 6$, larger particles damp flow dynamics notably and the SPH system cannot gain much speed-up, since the neighbour search then becomes a new bottleneck. Furthermore, a non-uniform SPH in combination with kernel averages [Mon92] is only stable for mass differences up to a factor of 10 [SP08] and non-uniform smoothing radii introduce larger errors if they differ by more than a factor 2 between particle neighbours [BOT01]. In the future, we would like to combine our blending of quantities with virtual boundary particles as proposed by Keiser et al. [KAG*06] in order to enable larger resolution differences. However, the presented temporal blending technique is independent from the smoothing kernels and refinement patterns. Please see the accompanying video for further results and for visual impressions of the presented scenes.

## 10. Conclusion

We have presented a novel temporal blending approach which is capable of exchanging particle sets while maintaining a consistent convection-diffusion simulation by using standard SPH rules only. Our temporal blending significantly reduces the influence of sampling errors while preserving the integration time-step. Additionally, we have introduced a scheme to control the blending time according to a predicted error in the pressure term.In order to evaluate the flexibility of our system, we have integrated our temporal blending into the latest approaches presented in the field of SPH-based fluid simulations. In combination with our new blending approach, our fully GPU-based implementation achieves interactive frame-rates for up to a million of particles.

## Acknowledgments

This work is partially funded by the Siegener Graduate School “Development of Integral Heterosensor Architectures for the n-Dimensional (Bio)chemical Analysis”. Also, we thank Maik Keller for his help in the video editing, the reviewers for their valuable comments and suggestions and the team of osgCompute for providing the GPU framework.

## 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).

[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.

[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), 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.

[GP11] GOSWAMI P., PAJAROLA R.: Time adaptive approximate SPH. In *Proceedings Eurographics Workshop on Virtual Reality Interaction and Physical Simulation* (2011).

[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.

[LD09] LENAERTS T., DUTRÉ P.: Mixing fluids and granular materials. *Comput. Graph. Forum 28*, 2 (2009), 213–218.

[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.

[Luc77] LUCY L. B.: A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82* (1977).

[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.

[OKK10] ORTHMANN J., KELLER M., KOLB A.: Topology-caching for dynamic particle volume raycasting. In *VMV* (2010), Eurographics Association, pp. 147–154.

[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.

[SB12] SCHECHTER H., BRIDSON R.: Ghost sph for animating water. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (2012).

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

[She68] SHEPARD D.: A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference* (1968), ACM, pp. 517–524.

[SP08] SOLENTHALER B., PAJAROLA R.: Density contrast sph interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), Eurographics Association, pp. 211–218.

[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.