Noname manuscript No.
(will be inserted by the editor)

Fast SPH Simulation for Gaseous Fluids

Bo Ren · Xiao Yan · Chen-feng Li · Tao Yang · Ming C. Lin · Shi-min Hu

Received: date / Accepted: date

Abstract This paper presents a fast SPH (Smoothed Particle Hydro-dynamics) simulation approach for gaseous fluids. Unlike previous SPH gas simulators, which solve the transparent air flow in a fixed simulation domain, the proposed approach directly solves the visible gas without involving the transparent air. By compensating the density and force calculation for the visible gas particles, we completely avoid the need of computational cost on ambient air particles in previous approaches. This allows the computational resources to be exclusively focused on the visible gas, leading to significant performance improvement of SPH gas simulation. The proposed approach is at least 10 times faster than the standard SPH gas simulation strategy, and is able to reduce the total particle number by 25-400 times in large open scenes. The proposed approach also enables fast SPH simulation of complex scenes involving liquid-gas transition, such as boiling and evaporation. A particle splitting and merging scheme is proposed to handle the degraded resolution in liquid-gas phase transition. Various examples are provided to demonstrate the effectiveness and efficiency of the proposed approach.

Keywords smoothed particle hydrodynamics · gas simulation · adaptive particle splitting and merging · phase transition

Bo Ren
first address
E-mail: renboeverywhere@gmail.com

Xiao Yan
Department of Computer Science and Technology, Tsinghua University, Beijing
E-mail: shiotoli@gmail.com

Chen-feng Li
College of Engineering, Swansea University
E-mail: c.f.li@swansea.ac.uk

Tao Yang
Department of Computer Science and Technology, Tsinghua University, Beijing
E-mail: yangtao9009@gmail.com

Ming C. Lin
Department of Computer Science, University of North Carolina at Chapel Hill
E-mail: lin@cs.unc.edu

Shi-min Hu
Department of Computer Science and Technology, Tsinghua University, Beijing
E-mail: shimin@tsinghua.edu.cn

1 Introduction

The smoothed particle hydrodynamics (SPH) method has become increasingly popular for simulating fluid motion for a wide range of natural phenomena and special effects. Yet, despite its popularity in liquid simulation, much less research has been reported in the literature on SPH simulation of gaseous fluids. Liquid is often considered to be incompressible in simulation, though gas usually behaves far more compressible. The time stepping condition in SPH methods requires much smaller time step simulating highly incompressible fluids than compressible fluids. This makes the SPH approach potentially more promising for gas simulation in a performance perspective. However, SPH simulation of gaseous fluids has not been sufficiently explored.

The challenge associated with fast SPH gas simulation can be better understood by examining a simple example, a plume of smoke rising in the air. Using a standard SPH simulator, the air flow is simulated within a fixed domain and the visible plume is represented by a density field that flows with the transparent air. The simulation domain is usually much larger in its extent than the space occupied by the actual evolving

plume; therefore the majority of the computational cost is spent on simulating the invisible air. The computational cost required to compute the air flow in a large simulation domain can be prohibitively high, making these approaches difficult, if not impossible, to use in simulating large, complex scenes with moving plumes, e.g. a steam train moving through a field or plumes of chimney smoke billowing in a village.

To address this issue, we propose a novel SPH scheme for gaseous fluids in which computation is performed only on the visible gaseous fluids, thus minimizing the simulation cost on ambient air gas particles. The interpolation error for the gas density is compensated for by estimating the average influence of the absent ambient air particles. The missing interaction between the simulation particles and the absent ambient air particles is also compensated for by a virtual pressure that is locally determined based on the current fluid dynamics and the atmosphere pressure. We demonstrate that such an approach is able to minimize the unnecessary computation on ambient gas particles, while significantly improving the runtime performance of SPH gas simulation. The new method also enables fast SPH simulation of complex scenes that involve liquid-gas phase transition. A robust particle splitting and merging strategy is proposed for liquid-gas transition situations; we show that phase transition phenomena can be efficiently captured in a unified SPH framework. We highlight the performance benefits (up to two orders of magnitude) of this approach on several large-scale phenomena as shown in Figs. 5, 4, 6, and 7.

2 Previous Work

Recent SPH methods [20,22,27], have focused primarily on liquid simulation. [29] proposed an SPH scheme to create and preserve vortical details near moving objects in gas simulation. However, they fill the whole simulation domain with particles. In some hybrid methods, SPH gas particles are generated in selected regions for various reasons, e.g. to reduce the computational cost of solving high-speed gas flow [11] and to help track the interface and compute gas-liquid interaction [6]. Nonetheless, the entire gas domain needs to be simulated by the grid-based simulator in these hybrid approaches to achieve high-accuracy simulation results.

SPH simulation generally simulates the entire domain because particle deficiencies lead to the erroneous interpolation of particle properties, such as density and pressure. To overcome this degradation of accuracy at rigid boundaries, SPH simulations often add solid particles, both to avoid particle penetration and to improve density computation of the liquid [19,8,5]. For water-air boundaries, ambient air particles have been added through a serialized sampling process to improve the density and surface tension calculation of the liquid [16, 25]. Compared to the interaction between water particles, the force between air and water is largely negligible and is not included in their liquid-simulation model. However, for gas simulation, the interaction between the visible plume and the surrounding ambient air is important and cannot be ignored.

Adaptive SPH simulations have previously used the split-and-merge technique for SPH particles. In [3,28, 14] SPH particles were bisected into different levels and particles at the same level were merged on demand. These methods achieve good results in liquid simulation, but require serialized adaptive sampling to maintain a uniform particle distribution and improve stability. More recently [26] proposed a two-scale particle scheme extending the processable density ratio, and [23] blended different levels of particles to achieve a smoother transition during splitting and merging. Their methods involve blending of different levels of particles and relaxing the clustered particles after splitting during a certain time period. This leads to better stability, but comes at an extra computational cost in the splitting process. Although these adaptive approaches adjust particle sizes to reduce the number of ambient air particles being simulated, the simulation cost still remains untenably high for very large simulation domains. Our approach is considerably faster than the previous methods that simulate ambient air particles, because it focuses the computing resources only on the visible gaseous fluids and estimating the average influence of the surrounding air. Such estimation is performed by compensating for the lost gas density and interaction forces that would be exerted by the unsimulated ambient air particles.

There are other works that are also partially related to our approach. [13] and [15] achieved large-scale liquid simulation with Lagrangian method. In [4], sparse particles were allowed in a vortex-based simulation scheme. [7] also tried to achieve similar goal of sparse representation using vortex sheet to represent smoke. Virtual particles for interface handling are also used in works involving solid simulation, such as in [21]. In [17,1] optimization methods are used to reduce simulation errors. In [12] the velocity field is decomposed and only a small part of it is simulated using high resolution grid, achieving faster simulation. Recently [18] proposed a unified representation focusing on simulating all types of fluids, in which constrains and weak cohesive forces are used to maintain particle density in cases of particle deficiency, however they involved neither long-ranged

Table 1 Definition of symbols in gas simulation.

Symbol	Meaning
$\bar{\rho}_i$	uncorrected density of particle i
ρ_i	corrected density of particle i
ρ_0	rest density of particle i
m_j	mass of particle j
$W(\mathbf{r}, h)$	smoothing kernel function
∇W_{ij}	short for $\nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h)$
$\mathbf{r}_i, \mathbf{r}_j$	position of the i-th, j-th particle
V_0	intermediate variable
p_i	pressure of particle i
T_i	temperature of particle i
$\mathbf{n}_i$	local normal vector at the particle i
P_a, C_d	user defined strength factors
C_b, D_c, D_r	user defined coefficients for buoyancy
C_N	user defined threshold value

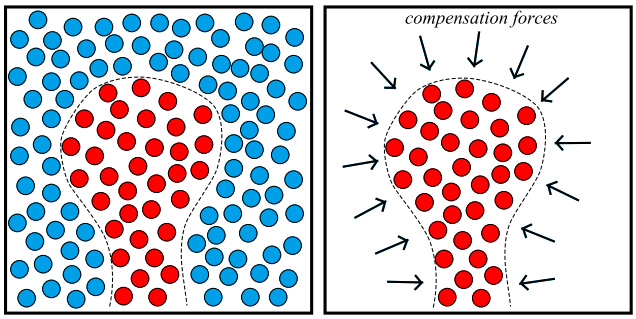
movement of smoke in large scenes nor phase transition in the work.

3 Fast SPH Gas Simulation

In gas simulation, standard SPH simulators (as shown in Fig. 1(a)) fill the entire simulation domain with ambient air particles. This is impractical for large or unbounded scenes, where large numbers of particles are needed to fill the simulation domain. Yet simply removing some of these particles causes particle deficiency. This is not an issue for free-surface liquids (liquid-air systems), where air particles are usually ignored without consequence, since the liquid and air densities differ so greatly. Atmospheric forces are negligible compared to the liquid particle interaction, and the missing air particles do not cause noticeable errors in calculation. These particle deficiencies do, however, pose serious problems for gaseous simulation with SPH. The forces between the plume and the surrounding air are not negligible as in liquid simulations, but are crucial in determining the movement of simulated gas particles.

In order to achieve high-fidelity visual effects without filling the entire space with ambient air particles, we must compensate for the accuracy degradation from particle deficiency. Particle deficiency has two main effects on accuracy: (1) erroneous density calculation of SPH particles and (2) missing interaction forces from the ambient air particles.

Following the same work flow as the standard SPH method, we derive below a fast SPH scheme (illustrated in Fig. 1(b)) which only simulates the visible gas and does not model the surrounding ambient air. We have developed appropriate compensation formulations for both gas-density and particle-force calculations. Instead of using a secondary density field to represent the distribution of the visible gas plume, we use the SPH gas particles to directly represent the visible gas; this is also convenient for phase transformation of particles in liquid-gas phase transitions.



(a) (b)

Fig. 1 (a) The standard approach fills the simulation domain with ambient air particles (blue). (b) Our approach removes these ambient air particles and instead adds compensation forces to the gas particles (red).

3.1 Gas density correction

Assuming the ambient air particles are included in the SPH simulation, the following relations hold for a given particle i:

$$\rho_i = \sum_j m_j W_{ij} + \sum_{j'} m_{j'} W ij' \tag{1}$$

$$0 = \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} + \sum_{j'} \frac{m_{j'}}{\rho_{j'}} \nabla W_{ij'} \tag{2}$$

where the index j refers to the SPH particles of interest, the index j' refers to the ambient air particles, ρ_s denotes the density of particle s, m_s denotes the mass of particle s, and $\nabla W_{st} = \nabla_s W(\mathbf{r}_s - \mathbf{r}_t, h)$ is the gradient of the smoothing kernel function with support h. We use the standard smoothing kernel functions as in [20]. Eqn. (1) and Eqn. (2) both come from the standard SPH formulation, where Eqn. (1) is the density interpolation and Eqn. (2) is a first-order identity that holds exactly for ideal particle distributions.

If the ambient air particles are simply removed from the simulation, the second term in Eqns. (1) and (2) will disappear, leading to erroneous results. To correct this error, it is assumed that a virtual "average" particle k can be placed nearby to satisfy Eqns. (1) and (2), i.e.:

$$\rho_i = \sum_j m_j W_{ij} + m_k W_{ik} \tag{3}$$

$$0 = \sum_j \frac{m_j}{\bar{\rho}_j} \nabla W_{ij} + \frac{m_k}{\rho_k} \nabla W_{ik} \tag{4}$$

In Eqns. (3) and (4) the effect of ambient air particles surrounding the particle i is compensated for by a virtual "average" particle k paired with it. We use $\bar{\rho}_i =$

$\sum_j m_j W_{ij}$ to denote the uncorrected density value, i.e. the density interpolation following Eqn. (1) without the second term. To determine the properties of the virtual particle k, it is first assumed that densities in adjacent particles change gradually in the gas simulation; the density of particle k is then considered to be the same as the interpolated density of particle i, i.e. $\rho_k = \rho_i$. Then the distance from any virtual particle k to its paired real particle i is set as a fixed value, which will be discussed later. Eqn. (4) contains two vector-valued functions that cancel each other. The first term in Eqn. (4) can be computed directly and is known for a given particle i. On the right-hand side of Eqn. (4), the direction of the second term must be in line with the first term. Therefore, we only need to consider the norms of the two terms in Eqn. (4), which can be rewritten as:

$$m_k = V_0 \rho_k = V_0 \rho_i \tag{5}$$

where $V_0 = \frac{|-\sum_j \frac{m_j}{\bar{\rho}_j} \nabla W_{ij}|}{|\nabla W_{ik}|}$ can be predetermined for particle i.

Substituting Eqn. (5) into Eqn. (3), the corrected density of particle i can be computed as:

$$\rho_i = \frac{\bar{\rho}_i}{1 - V_0 W_{ik}} \tag{6}$$

Eqn. (6) corrects the density of the SPH particles in the absence of surrounding ambient air particles. However, for a more accurate evaluation, the densities of adjacent particles in Eqn. (4) should also use the corrected value ρ_j instead of the uncorrected one $\bar{\rho}_j$. This leads to the following formulae, whose detailed proof is given in Appendix A:

$$m_k = V_0 \bar{\rho}_i \tag{7}$$

$$\rho_i = \bar{\rho}_i (1 + V_0 W_{ik}) \tag{8}$$

In the above calculation, the distance from a virtual particle k to its paired real particle i is heuristically set as a fixed value δh, where $\delta \in [0.4, 0.7]$ usually brings more plausible results in our experiments. There may be other methods that allow this distance to vary, but we have found simply fixing the distance produces sufficiently good results for the SPH gas simulation.

3.2 Particle force compensation

After removing the ambient air particles from the simulation, the interaction forces between the simulation particles and the ambient air particles must also be compensated for. The missing interaction forces are responsible for three physical effects: the local force balance, the atmospheric pressure and the local gas viscosity. Each of these physical effects requires that an appropriate compensation be made.

The virtual particles generated for the gas density correction can be used to compensate the local force balance. Following the standard SPH formulation, the pressure force for a given particle i is:

$$\mathbf{F}_{p,i} = -\sum_j \frac{m_j}{\rho_j} \frac{p_i + p_j}{2} \nabla W_{ij} \tag{9}$$

where p_s is the gas pressure calculated at particle s. Using the paired virtual particle k, an extra term is added to Eqn. (9):

$$\tilde{\mathbf{F}}_{p,i} = -\sum_j \frac{m_j}{\rho_j} \frac{p_i + p_j}{2} \nabla W_{ij} + \frac{m_k}{\rho_k} \frac{p_i + p_k}{2} \nabla W_{ik} \tag{10}$$

Here the interpolated density uses the corrected density computed from Eqn. (8), and the pressure is computed from the ideal gas state function:

$$p = \kappa(\rho - \rho_0) \tag{11}$$

where κ is the gas constant, and ρ_0 is the gas density at rest.

The atmospheric pressure pushes the gas towards its inner direction; to simulate this effect, we introduce a normal force. First, the local normal vector at the particle i is computed as:

$$\mathbf{n}_i = \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \tag{12}$$

The normal vector defined above is identical to the first term in Eqn. (2). For those internal simulation particles whose neighborhoods do not contain any ambient air particles, the second term in Eqn. (2) is zero, and therefore the norms of their normal vectors are zero or approximately zero, depending on the particle distribution. For those simulation particles that are adjacent to the missing ambient air particles, the second term in Eqn. (2) is non-zero, and therefore the norms of their normal vectors are non-zero as defined by the first term. Based on this analysis, the atmospheric pressure force is defined as:

$$\tilde{\mathbf{F}}'_{p,i} = P_a \mathbf{n}_i = P_a \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \tag{13}$$

where P_a is a user-defined strength coefficient reflecting the ambient air pressure. The pressure force defined above automatically ensures that the atmospheric pressure is added only to the particles on the gas boundary, since the inner gas particles have zero norms for their normal vectors.

To account for the effect of the local gas viscosity, we introduce a damping effect proportional to the particle velocity as follows:

$$\tilde{\mathbf{a}}_{damp,i} = -C_d \mathbf{v}_i \tag{14}$$

where $\mathbf{a}_{damp,i}$ is the acceleration of the i-th particle due to the damping force, C_d is the damping coefficient,

and $\mathbf{v}_i$ denotes the velocity of particle i. This damping force is applied only if the norm of the particle normal exceeds a given threshold value, i.e. $|\mathbf{n}_i| > C_N$. This is to ensure that the damping effect is only added to those boundary particles with neighborhood deficiency.

3.3 Buoyancy and boundary treatment

The buoyancy force is related to the temperature of SPH gas particles. Similarly to [2], the temperature of each gas particle is evolved as:

$$\frac{dT_i}{dt} = \sum_j \frac{m_j}{\rho_i \rho_j} D_c (T_i - T_j) \frac{(\mathbf{r}_i - \mathbf{r}_j) \cdot \nabla W_{ij}}{(\mathbf{r}_i - \mathbf{r}_j)^2 + \gamma^2} \tag{15}$$

where T_s is the temperature of particle s, D_c is the thermal conductivity, and $\gamma^2 \ll 1$ is a small positive number to avoid computational singularity. For those particles with $|\mathbf{n}_i| > C_N$, a radiation term is also applied to simulate the heat transfer to the missing particles:

$$\frac{dT_i}{dt} = -T_i / D_r \tag{16}$$

where D_r denotes the radiation half time, lower value of D_r leads to faster cooling of the corresponding particles. Then the buoyancy of the gas particle is set to be proportional to the temperature, i.e.

$$\tilde{\mathbf{a}}_{b,i} = C_b T_i \mathbf{b} \tag{17}$$

where $\tilde{\mathbf{a}}_{b,i}$ is the acceleration of the i-th particle due to the buoyancy force, C_b is the buoyancy coefficient and $\mathbf{b}$ is a unit vector pointing to the buoyancy direction.

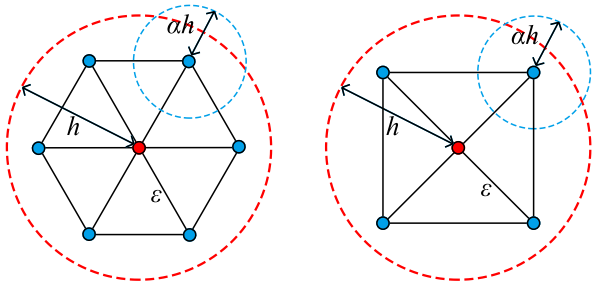
In this work, the rigid boundaries are represented with boundary particles. The positions of boundary particles are not affected by gas particles, but they participate in all SPH calculations in the same way as normal simulation particles, as in [27].

4 Phase Transition between Liquid and Gas

Using the proposed fast SPH scheme for gas simulation, it becomes possible to develop an efficient and unified SPH scheme for complex scenes that involve liquid-gas phase transition. The density of liquid is often 1000 times higher than the gas density. When the phase transition occurs, this high density ratio causes a drop in simulation resolution and robustness, and ultimately loss of visual plausibility. To prevent this loss of plausibility, this situation requires a proper particle splitting and merging strategy, which is explained in the following sections.

4.1 Keeping simulation resolution in phase transition

Due to the large density ratio between liquid and gas phases, the effective volume of an SPH particle will be significantly larger in the gas phase if it is directly vaporized from the liquid phase. This leads to degraded simulation resolution. To overcome this problem, we split the involved liquid particles before transferring them into the gas phase. To be truthful to the underlying physics, the change of local mass distribution due to this refinement process must be minimized, and the global mass and momentum conservation should be preserved. We propose a pattern-based splitting scheme originated from [10] to achieve this goal.



(a) (b)

Fig. 2 2D examples of particle splitting. Based on a predefined pattern, e.g. the hexagon pattern shown in (a) and the square pattern shown in (b), the mother particle (the red node) is split into a set of evenly distributed daughter particles (the blue nodes). The distance between the mother particle and the daughter particles is ϵ. The smoothing radius of the daughter particles is αh, where h denotes the smoothing radius of the mother particle.

As shown in Fig.2, the mother particle (denoted by the red node) is split into a set of daughter particles (denoted by the blue nodes), which are evenly distributed around the original position. Depending on the simulation need (to be explained in Section §4.2), a daughter particle can also be placed at the centre, i.e. the position of the mother particle. The distance from the mother particle to each daughter particle is denoted by a scalar parameter ϵ. As in [10,3], we allow variable smoothing lengths in the simulation. Thus, another scalar parameter α is introduced to denote the change of smoothing radius, i.e. $h_{new} = \alpha h_{origin}$. Finally, the mass ratio between the daughter particles and the mother particle is denoted by a vector $\lambda = [\lambda_1, ..., \lambda_M]^T$, where $i = 1, 2, \cdots, M$ denotes the daughter particles and M denotes the number of daughter particles. Thus, the mass of each daughter particle is $m_i = \lambda_i m_N$, where N denotes the mother particle and m_N is the mass of the mother particle.

It is proved in [10] that, given the values α and ϵ, the refinement error can be written in terms of λ as

$$E[\alpha,\epsilon](\lambda) = \frac{m_N^2}{h_N^d}(\bar{C} - 2\lambda^T\bar{\mathbf{b}} + \lambda^T\bar{\mathbf{A}}\lambda) \tag{18}$$

where each $\bar{A}_{ij} = \frac{1}{\alpha^d}\int_\Omega W_i(\mathbf{r},1)W_j(\mathbf{r},1)d\mathbf{r}$, each $\bar{b}_j = \int_\Omega W_N(\mathbf{r},1)W_j(\mathbf{r},1)d\mathbf{r}$, $\bar{C} = \int_\Omega W_N^2(\mathbf{r},1)d\mathbf{r}$, and d is the simulation dimension. $W_i(\mathbf{r},1) = W(\mathbf{r}-\mathbf{r}_i,1)$ denotes the smoothing kernel function at the i-th particle position $\mathbf{r}_i$ with the unit kernel radius.

Therefore, given the refinement parameters (ϵ,α), the optimal values for λ^* can be obtained by minimizing the error $E[\alpha,\epsilon]$ with the constraint of $\sum_{j=1}^M \lambda_j = 1$. Note that in a given pattern, for each choice of (ϵ,α), the coefficients $\bar{C}$, $\bar{\mathbf{b}}$, $\bar{\mathbf{A}}$ can be calculated numerically, which allows for the pre-computation of the best λ^*. This process can also be reversed, and the corresponding (ϵ,α) can be found from a desired λ^* pattern. For a certain pattern, at the precomputation stage we need to calculate corresponding values of (E,λ) from (ϵ,α) pairs and save them into a two-dimension chart. This provides the necessary information to determine better patterns that give a good balance between refinement error and mass distribution.

4.2 Boiling and evaporation

Liquid can transform into gas through boiling or vaporizing. The former is a vibrant process occurring at the boiling point, and the latter is a quiet process taking place slowly at lower temperatures. The pattern-based splitting scheme in §4.1 provides a natural way to model these two different phenomena. For boiling we choose a splitting pattern that does not place a daughter particle at the center. Patterns without a central node usually divide the mass of the original particle evenly into new particles, especially when the pattern is symmetric. Typically we use a regular icosahedron pattern with $(\epsilon,\alpha) = (0.3, 0.9)$, but other patterns can also be chosen when necessary. After splitting, the new particles are transferred into the gas phase, experiencing density drop and volume expansion in the boiling phenomena. The smoothing radius of these gas particles is also enlarged in this process, matching their expansion in effective volume. For the relatively slower vaporization, we choose a splitting pattern with a daughter particle in the center. In these types of patterns, the majority mass of the original particle remains in the new particle at the center. For example, in a cubic pattern with $(\epsilon,\alpha) = (0.4, 0.9)$ and with a relatively low refinement error, the mass of the new particle is $\lambda_i = 0.992$ for the particle at the center and is $\lambda_i = 0.001$ for others. After particle splitting, the center particle remains in the liquid phase and the other particles are transformed into the gaseous phase. The smoothing radius of these transformed particles is also re-adjusted.

The above patterns are chosen with balanced consideration. The cubic pattern, for example, provides a moderate phase transition rate between liquid and air, with only a small fraction of liquid mass turned into gas phase in each splitting, and keeps the effective volume of the gas particles similar to that of the liquid particles. For the boiling process the regular icosahedron pattern provides a good balance between simulation efficiency (i.e. avoiding particle number explosion) and resolution. These patterns also provide a relatively small refinement error under the above mass distribution. There may be better patterns, however the symmetric and easy-to-compute property of these patterns make the calculation both in precomputation and actual simulation significantly easier and faster. The effect of refinement error related to different patterns and corresponding (ϵ,α) values, and the effectiveness of this method comparing to original distribution are thoroughly analyzed and validated in [10].

In evaporation, a liquid particle could be split several times. It is possible to pre-compute different splitting schemes with different (ϵ,α) values for each split, making the mass of the new gas particles similar to each other. However, after several splits, the mass of the liquid particle reduces and the splitting error discussed in §4.1 becomes gradually larger. To overcome this problem, we split liquid particles only up to a threshold number of splits. Once a liquid particle reaches this splitting threshold, we merge the particle with adjacent similar liquid particles if possible.

Specifically, for each candidate particle M, we compute its inertia matrix as

$$\mathbf{I}_M = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{pmatrix} \tag{19}$$

where $I_{xx} = \sum_j m_j((y_j - y_M)^2 + (z_j - z_M)^2)$, $I_{yy} = \sum_j m_j((x_j - x_M)^2 + (z_j - z_M)^2)$, $I_{zz} = \sum_j m_j((x_j - x_M)^2 + (y_j - y_M)^2)$, $I_{xy} = \sum_j m_j(x_j - x_M)(y_j - y_M)$, $I_{yz} = \sum_j m_j(y_j - y_M)(z_j - z_M)$, and $I_{xz} = \sum_j m_j(x_j - x_M)(z_j - z_M)$. Here, the summation is performed over the neighborhood of the candidate particle M defined by its smoothing radius. Then, following [9], we define the mass decentration as:

$$\eta_M = |(\frac{trace(I_M)}{3})^3 - det(I_M)| \tag{20}$$

The smaller the η_M value is, the mass distribution within the neighborhood of particle M is more similar to a uniform sphere. The mass decentration η_M vanishes

when the mass distribution around the candidate particle M is an ideal sphere. In our implementation, the particle with the smallest mass dencentration is merged with its neighborhood. The merged particle, denoted by M', has the total mass of the entire neighborhood before merging, and it is placed at the same position as M. After merging, the merged particle M' is split again into several particles of proper size, following the splitting algorithm described in §4.1. As shown in Fig. 3, this merge-split process ensures the quality of SPH particles during merging and splitting operations.

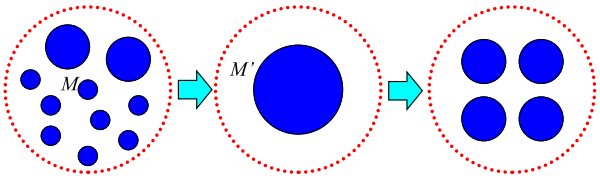


Fig. 3 (a) The candidate particle M to be merged with its neighborhood. (b) A larger particle M' is generated after merging. (c) Then the large particle M' is split into a set of daughter particles with desired size and distribution.

It is noted that, right after the splitting or merging process, partially "overlapping" daughter particles can temporarily appear. However the new particle position, mass, etc. are carefully pre-computed so that particle interactions in the new distribution are maintained almost the same as before even in the case of partial overlapping. Thus stable simulation can be achieved without additional calculation like some other methods, e.g. Poisson disc sampling.

5 Implementation

In the standard SPH algorithm, the motion of a particle i is determined by

$$\rho_i = \sum_j m_j W_{ij} \tag{21}$$

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{1}{\rho_i}\mathbf{F}_{p,i} + \frac{1}{\rho_i}\mathbf{F}_{v,i} + \mathbf{g} \tag{22}$$

where $\mathbf{F}_{p,i}$ denotes the pressure force, $\mathbf{F}_{v,i}$ the viscosity force, $\mathbf{g}$ the gravity acceleration, $\mathbf{a}_i$ the acceleration of particle i, $\mathbf{v}_i$ the velocity of particle i, and t the time variable. Our approach first compensates for the density of each particle using Eqn. (8). Then the local pressure balance and the atmospheric pressure are combined to form the pressure force term, i.e.

$$\mathbf{F}_{p,i} = \tilde{\mathbf{F}}_{p,i} + \tilde{\mathbf{F}}'_{p,i} \tag{23}$$

where $\tilde{\mathbf{F}}_{p,i}$ and $\tilde{\mathbf{F}}'_{p,i}$ are defined in Eqns. (10) and (13).

The damping and buoyancy accelerations defined in Eqns. (14) and (17) are also added to the right-hand side of Eqn. (22). The complete equation of our approach is:

$$\mathbf{a}_i = \frac{1}{\rho_i}(\tilde{\mathbf{F}}_{p,i} + \tilde{\mathbf{F}}'_{p,i}) + \frac{1}{\rho_i}\mathbf{F}_{v,i} + \mathbf{g} + \tilde{\mathbf{a}}_{damp,i} + \tilde{\mathbf{a}}_{b,i} \tag{24}$$

To obtain more visual variance of the simulated gas, it is possible to artificially adjust the atmospheric pressure $\tilde{\mathbf{F}}'_{p,i}$ according to the particle's normal vector. Specifically, $\tilde{\mathbf{F}}'_{p,i}$ is multiplied by a constant factor $\beta \in [0,1]$ if the norm of its normal vector is less than a given threshold $|\mathbf{n}_i| < C_N$, where near-zero β values will offer more visual variation.

If desired one can also add artificial vortex to the simulation, such as Lagrangian vortex methods adopted in [24,18]. Specifically, the vortex carried by each particle is evolved by the following equation:

$$\frac{D\omega_i}{dt} = \omega_i \cdot \nabla\mathbf{v} + \beta(\mathbf{n}_i \times \mathbf{g}) \tag{25}$$

where ω_i is vortex carried by the i-th particle, β is a user-defined constant used to control the overall strength of the effect. Then a particle is driven by an extra force due to the neighbors:

$$\mathbf{F}_{vort,i} = \sum_j (\omega_j \times (\mathbf{r}_i - \mathbf{r}_j))W_{ij} \tag{26}$$

The above vortex method adds more visual variance to the result with about 15% increase in computational cost.

6 Result

The performance gain for the proposed approach will depend on the size of the simulation domain, in that larger domains require more ambient air particles to fill the entire space. Our approach is most efficient dealing with scenes that have smoke plumes evolving across a large empty space with open boundaries, which we'll demonstrate in the examples. For rendering, the fluid particles are directly rendered, which is convenient for liquid-gas phase transition, and it can easily cope with long-range movement of gaseous fluids in very large scenes. Post-processing rendering techniques, similar to those in [18], can also be adopted to enhance diffusion effects, which we use in the steam train example.

To demonstrate the effectiveness and speed up of our approach, a boiling example is presented in Fig. 4. At the center of the scene, the water is boiled in a small pan producing vapor. In Fig. 4(a), using our approach described in §3 and §4, the vapor phase is successfully simulated, showing plausible movement and shape.

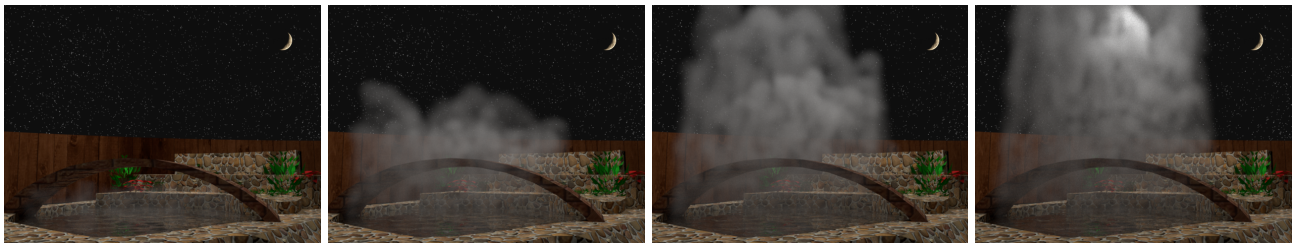
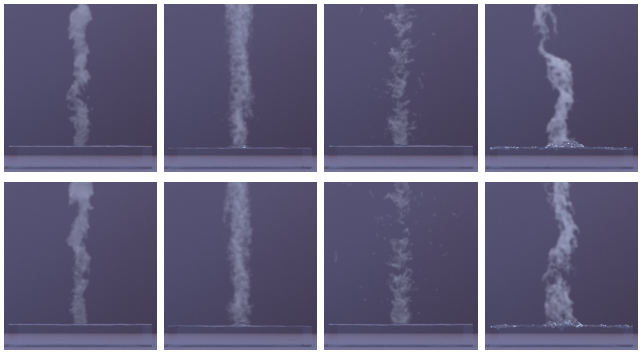


Fig. 5 Evaporation. A misty fog rises from an outdoor bath. The simulation includes about 407,000 particles in total, which is approximately 25 times faster than filling the scene using an estimated 9.9 million ambient air particles with standard approach.



(a) (b) (c) (d)

Fig. 4 Boiling. (a) Our approach. Using only about 9,000 vapor particles in total, the simulation runs at 67.3 steps per second. (b) Standard approach with ambient air particles. 640,000 ambient air particles are required to fill the empty space, reducing the simulation speed to 7.4 steps per second. (c) Removing the correction and compensation from (a), the simulation speed is comparable with that in (a) at 84.6 steps per second, however the vapor does not hold its shape, scattering and becoming thin. (d) An implementation of the two-scale particle simulation method. Up to 305,000 particles are generated in the boundary region of the method. The simulation speed is 4.9 steps per second.

The simulation uses only about 9,000 vapor particles in total, plus about 40,000 liquid particles, and with artificial vortex added the simulation runs at 67.3 steps per second on a Nvidia GeForce GTX 780 GPU. The rendering volume grid is $160 \times 120 \times 400$. The simulation can run stably at 2 steps per frame (step size 1/60s). For comparison, a vapor plume with the same resolution, but simulated using fully-filled ambient air particles, is shown in Fig. 4(b). In this case, no correction or compensation is needed, but this small, simple scene requires about 640,000 ambient air particles to fill the empty space, reducing the simulation speed to 7.4 steps per second. In Fig. 4(c), we demonstrate what happens when we remove the the correction and compensation from the case shown in Fig. 4(a). The erroneous calculation causes the vapor phase to scatter and become thin. The simulation speed is comparable with (a) at 84.6 steps per second.

Previous SPH methods using adaptive-sampling have mainly dealt with liquid simulation; it is not straightforward to extend previous methods to GPU-based gas simulation without adjustment. We tested the method of [Solenthaler and Gross 2011] on the above example. The vapor particles are used to define the active region, and the low-resolution simulation in their method is carried on by a simulation of half resolution. The result is shown in Fig. 4(d). Their method still needs to fill the entire simulation domain with ambient air particles, and a large amount of particles are also needed in the boundary region. In total, the low-resolution simulation contains about 88,000 particles (including liquid particles), and up to 305,000 "boundary particles" are added into the simulation around the vapor particles; the simulation speed is 4.9 steps per second. This speed is slower than the standard algorithm, possibly because the splitting and relaxing algorithm for the generation of "boundary particles" are not as well parallelizable on GPU as other parts of the implementation, and because the number of "boundary particles" needed does not give much performance improvement space in gas simulations.

Fig. 6 shows a steam train running across open terrain and passing through a tunnel in a light-yellow hill. We simulate the smoke rising from its smokestack in a $200 \times 420 \times 1920$ region measured by volume data grid size in rendering. If the standard approach were used, the example would have required an estimated 119.3 million ambient air particles in order to fill the simulation space. In this case the artificial vortex is added. With our approach only 433,000 gas particles are used in total, and the simulation is able to run at 17.0 steps per second, achieving more than 200 times speed up.

Fig. 5 shows an outdoor bath using a rendering volume grid of $260 \times 260 \times 400$, where the water is evaporating and producing a misty fog. The gas phase has a rest density that is 1/1000 of the liquid phase. Filling this scene would need 9.9 million ambient air particles besides liquid particles. With our approach, up to 329,000 particles are used to simulate the vapor plus about 78,000 liquid particles, achieving approximately 25 times speed up. The simulation runs at 13.9 steps per second.

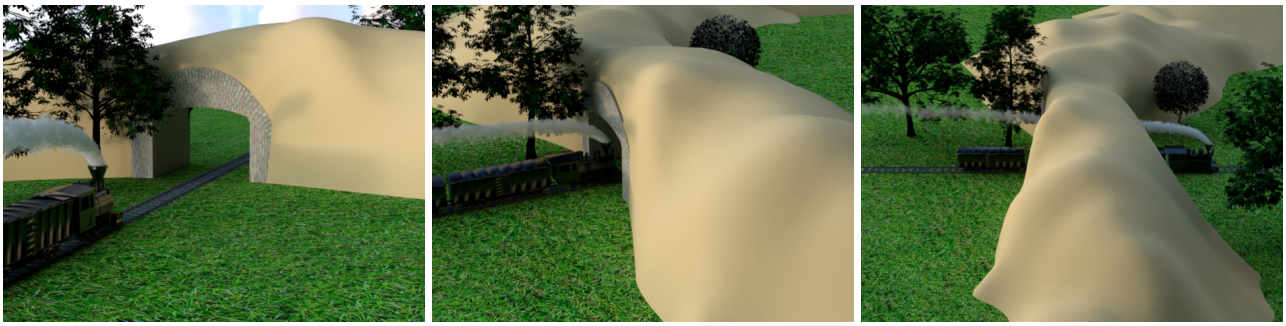


Fig. 6 Steam train running across an open terrain through a tunnel in a light-yellow hill. The smoke rising from its smokestack is simulated with up to 433,000 gas particles using our approach. If ambient air particles were used, this example would have required an estimated 119.3 million ambient air particles in order to fill the simulation space. Our approach achieves more than 200 times speed up on this scene.

Fig. 7 Plumes rising from an ancient city towards a magic attractor located above the center building. Up to 1.3 million particles are used with our approach, saving computational cost on more than 600 million ambient air particles. More than 400 times speed up is achieved in this scene.

Fig. 7 shows an ancient city, from which a number of plumes are rising into the sky towards a magic attractor located above the center building. The plumes then converge and mix around the magic attractor. Based on the proposed approach, up to 1.3 million particles are used in this example and the simulation runs at 3.0 steps per second. The rendering volume grid of this example reaches $1320 \times 560 \times 1240$. If the standard approach were used, the example would have required more than 600 million ambient air particles in order to fill the simulation space. For such a large scene it would be too costly for normal GPUs to simulate ambient air particles, and our approach brings more than 400 times speed up to its simulation.

7 Conclusion

We have developed a novel fast SPH simulation approach for gaseous fluids. The proposed approach is able to completely avoid simulating ambient air particles, bringing significant performance benefit (up to two orders of magnitude) to SPH gas simulation, especially for large open scenes. Liquid-gas phase transition phenomena, such as boiling and evaporation, can also be efficiently captured following the fast simulation approach.

Unlike grid-based solvers, the SPH schemes do not have built-in diffusion, and it is difficult to handle the diffusion effect using the proposed approach. Although it can be partially alleviated by rendering, this is an intrinsic limitation related to single-phase SPH gas simulation, and using multi-fluid models can be a possible research direction in the future. Removing ambient air particles also makes it difficult to analyze the momentum conservation property. Although little disobedience of momentum conservation is visually observed for the overall gas movement, the global momentum conservation is not accurately ensured in the proposed approach. Theoretically this could lead to error accumulation, making the simulation possibly biased from the natural movement in very long simulations. Though the interactions between particles due to pressure is reproduced in our approach, the immediate viscous interactions between simulated particles and ambient air particles are not fully recovered, losing vortical motions to some degree. This can be partially compensated by adding artificial vortex, however additional compen-

sation mechanisms can be desirable. In reality, in relatively narrow spaces sometimes it is possible for the unseen air to "carry over" interactions between separated parts of visible gas through propulsion, viscosity or other mechanisms. However, since these ambient air particles are removed in our approach, these indirect interactions are no longer captured; our approach is thus less effective for simulating narrow surroundings.

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive comments, and Prof. Jianjun Zhang and Prof. Jian Chang for their suggestions. This work is supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Number 61120106007) and and the National High Technology Research and Development Program of China (Project Number 2012AA011503). The authors would also like to acknowledge the financial support provided by the National Science Foundation (NSF IIS-1320644) and UNC Carolina Development Foundation, and Sêr Cymru National Research Network in Advanced Engineering and Materials.

References

1.
2. Modelling confined multi-material heat and mass flows using SPH. Applied Mathematical Modelling **22**(12), 981 – 993 (1998). DOI http://dx.doi.org/10.1016/S0307-904X(98)10031-8
3. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. In: ACM Transactions on Graphics (TOG), vol. 26, p. 48. ACM (2007)
4. Angelidis, A., Neyret, F., Singh, K., Nowrouzezahrai, D.: A controllable, fast and stable basis for vortex based smoke simulation. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '06, pp. 25–32. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006). URL http://dl.acm.org/citation.cfm?id=1218064.1218068
5. Bender, J., Erleben, K., Teschner, M., et al.: Boundary handling and adaptive time-stepping for pcisph. In: Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2010)
6. Boyd, L., Bridson, R.: Multiflip for energetic two-phase fluid simulation. ACM Trans. Graph. (2), 16
7. Brochu, T., Keeler, T., Bridson, R.: Linear-time smoke animation with vortex sheet meshes. In: J. Lee, P.G. Kry (eds.) Symposium on Computer Animation, pp. 87–95. Eurographics Association
8. Colagrossi, A., Landrini, M.: Numerical simulation of interfacial flows by smoothed particle hydrodynamics. Journal of Computational Physics **191**(2), 448–475 (2003)
9. Desbrun, M., Cani, M.P.: Space-Time Adaptive Simulation of Highly Deformable Substances. Rapport de recherche RR-3829, INRIA (1999). URL http://hal.inria.fr/inria-00072829
10. 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), 295–324 (2007)
11. Gao, Y., Li, C.F., Hu, S.M., Barsky, B.A.: Simulating gaseous fluids with low and high speeds. Computer Graphics Forum **28**(7), 1845–1852 (2009). DOI 10.1111/j.1467-8659.2009.01562.x. URL http://dx.doi.org/10.1111/j.1467-8659.2009.01562.x
12. Gao, Y., Li, C.F., Ren, B., Hu, S.M.: View-dependent multiscale fluid simulation. Visualization and Computer Graphics, IEEE Transactions on **19**(2), 178–188 (2013). DOI 10.1109/TVCG.2012.117
13. Gerszewski, D., Bargteil, A.W.: Physics-based animation of large-scale splashing liquids. ACM Trans. Graph. **32**(6), 185:1–185:6 (2013). DOI 10.1145/2508363.2508430. URL http://doi.acm.org/10.1145/2508363.2508430
14. Hong, W., House, D.H., Keyser, J.: Adaptive particles for incompressible fluid simulation. The Visual Computer **24**(7-9), 535–543 (2008)
15. Ihmsen, M., Cornelis, J., Solenthaler, B., Horvath, C., Teschner, M.: Implicit incompressible sph. IEEE Transactions on Visualization and Computer Graphics **20**(3), 426–435 (2014). DOI 10.1109/TVCG.2013.105. URL http://dx.doi.org/10.1109/TVCG.2013.105
16. Keiser, R.: Multiresolution particle-based fluids (2006)
17. Kim, S.T., Hong, J.M.: Visual simulation of turbulent fluids using mls interpolation profiles. The Visual Computer (12), 1293–1302
18. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Transactions on Graphics (TOG) **33**(4), 104 (2014)
19. Monaghan, J.J.: Simulating free surface flows with sph. Journal of computational physics **110**(2), 399–406 (1994)
20. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '03, pp. 154–159. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003). URL http://dl.acm.org/citation.cfm?id=846276.846298
21. Müller, M., Schirm, S., Teschner, M., Heidelberger, B., Gross, M.: Interaction of fluids with deformable solids: Research articles. Comput. Animat. Virtual Worlds **15**(3-4), 159–171 (2004). DOI 10.1002/cav.v15:3/4. URL http://dx.doi.org/10.1002/cav.v15:3/4
22. Müller, M., Solenthaler, B., Keiser, R., Gross, M.: Particle-based fluid-fluid interaction. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '05, pp. 237–244. ACM, New York, NY, USA (2005). DOI 10.1145/1073368.1073402. URL http://doi.acm.org/10.1145/1073368.1073402
23. Orthmann, J., Kolb, A.: Temporal blending for adaptive sph. In: Computer Graphics Forum, vol. 31, pp. 2436–2449. Wiley Online Library (2012)
24. Pfaff, T., Thuerey, N., Gross, M.: Lagrangian vortex sheets for animating fluids. ACM Trans. Graph. **31**(4), 112:1–112:8 (2012). DOI 10.1145/2185520.2185608. URL http://doi.acm.org/10.1145/2185520.2185608
25. Schechter, H., Bridson, R.: Ghost sph for animating water. ACM Transactions on Graphics (TOG) **31**(4), 61 (2012)
26. Solenthaler, B., Gross, M.: Two-scale particle simulation. In: ACM Transactions on Graphics (TOG), vol. 30, p. 81. ACM (2011)
27. Solenthaler, B., Pajarola, R.: Density contrast sph interfaces. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '08, pp. 211–218. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2008). URL http://dl.acm.org/citation.cfm?id=1632592.1632623

28. Yan, H., Wang, Z., He, J., Chen, X., Wang, C., Peng, Q.: Real-time fluid simulation with adaptive sph. Computer Animation and Virtual Worlds **20**(2-3), 417–426 (2009)
29. Zhu, B., Yang, X., Fan, Y.: Creating and preserving vortical details in sph fluid. Comput. Graph. Forum (7), 2207–2214

Appendix A: Derivation of Eqns. (7-8)

This appendix shows the detailed derivations of Eqn. (7) and Eqn. (8). Since densities in adjacent particles are assumed to change gradually in the gas simulations, we can further consider that for a particle i, the densities of its neighboring particles $\rho_{i,j}$ are corrected with the same proportion as itself and in a similar form as Eqn. (6):

$$\rho_{i,j} = \frac{\bar{\rho}_{i,j}}{1 - VW_{ik}} \tag{27}$$

Here we want to determine the value of the scalar V, which for a given particle i does not vary with adjacent particle j. Using the form in Eqn. (27), Eqn. (4) can be rewritten as

$$0 = (1 - VW_{ik}) \sum_j \frac{m_j}{\bar{\rho}_j} \nabla W_{ij} + \frac{m_k}{\rho_i} \nabla W_{ik} \tag{28}$$

Utilizing V_0 in Eqn. (5), the above equation can be further rewritten as:

$$m_k = (1 - VW_{ik}) V_0 \rho_i \tag{29}$$

Note that it has been assumed in the first place that the final correction form should be in the form of Eqn. (27), which is equivalent to requiring $m_k = V\rho_i$. Comparing this with Eqn. (29), the following equation can be obtained:

$$(1 - VW_{ik}) V_0 = V \tag{30}$$

Solving Eqn. (30) yields

$$V = \frac{V_0}{1 + V_0 W_{ik}} \tag{31}$$

Substituting Eqn. (31) into Eqn. (27) and Eqn. (29) leads to Eqns. (7-8).