9-2011

An efficient adaptive vortex particle method for real-time smoke simulation

Shengfeng HE
Singapore Management University, shengfenghe@smu.edu.sg

Hon-Cheng WONG

Un-Hong WONG

2011 12th International Conference on Computer-Aided Design and Computer Graphics

An Efficient Adaptive Vortex Particle Method for Real-Time Smoke Simulation

Shengfeng He[1], *Hon-Cheng Wong[1,2], Un-Hong Wong[2]
1. Faculty of Information Technology
2. Space Science Institute
Macau University of Science and Technology
Macao, China
*Email: shengfeng_he@yahoo.com, *hcwong@ieee.org, uhwong@must.edu.mo*
**Corresponding author*

Abstract

Smoke simulation is one of the interesting topics in computer animation and it usually involves turbulence generation. Efficient generation of realistic turbulent flows becomes one of the challenges in smoke simulation. Vortex particle method, which is a hybrid method that combines grid-based and particle-based approaches, is often used for generating turbulent details. However, it may cause irrational artifacts due to its initial condition and vorticity forcing approach used. In this paper, a new vorticity forcing approach based on the spatial adaptive vorticity confinement is proposed to address this problem. In this approach, the spatial adaptive vorticity confinement force varies with helicity, leading to the fact that the grid-based simulation driven by the vortex particle is now based on the velocity field. Furthermore, we introduce an adaptive vortex particle approach to improve the computational efficiency of the simulation by making the influencing region adapt with the velocity and eliminating those particles with zero velocity in the vorticity forcing method. A parallel smoke simulator integrating our approaches has been implemented using GPUs with CUDA. Experimental results demonstrate that our proposed methods are efficient and effective for real-time smoke simulation.

1. Introduction

Modeling and rendering of natural scenes are still challenging tasks in computer graphics. Fluid simulation is widely applied not only in academy, but also in the field of entertainment and film industry. As the requirement of realistic visual natural effects is increasing, achieving more realistic visual effects of fluid simulation is a target that researchers always struggle for. Large-scale components of motion are well captured in low-resolution simulation, however, increasing the grid size to capture small scale details will cause a scalability problem. Many methods have been proposed to address this problem over the past few years. Some of them [1], [2], [3] recovered these details by adding noise. But these models introduced a certain extent of computational complexity and could not achieve highly turbulent effects. In this paper, we are focusing on achieving real-time smoke effects with significant turbulent flows.

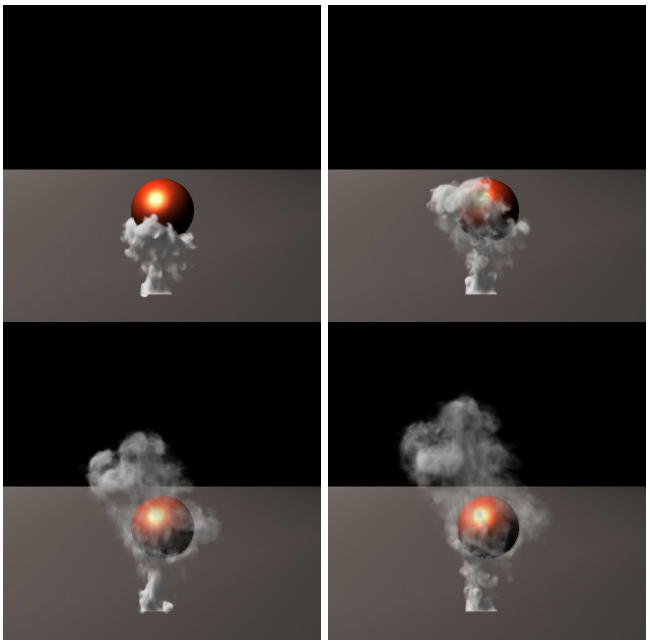

Figure 1. A set of sequential figures showing the smoke rising around a sphere. The simulation was done on a $256 \times 256 \times 256$ grid up-sampled from a $128 \times 128 \times 128$ coarse grid using 8192 particles based on the techniques proposed in this paper. Ray-marching was used in this simulation.

Vortex particle method [4] is a class of Lagrangian numerical methods for the simulation of incompressible fluid flows where the vorticity is an integral component. It is a hybrid method that overcomes the shortcoming of grid-based and particle-based methods, however, reconstructing the velocity field from particles is a tough work in this method. Selle *et al.* [5] presented a simple approach that orients to computer graphics and achieved highly turbulent

978-0-7695-4497-7/11 $26.00 © 2011 IEEE
DOI 10.1109/CAD/Graphics.2011.69

effects. In order to avoid computing the velocity from the particle vorticity, this method uses the grid-based velocity field as the velocity of particles. Furthermore, a simple and fast vorticity forcing approach — vorticity confinement is used. Vorticity confinement [6] was proposed to recover small rolling details. However, it is not stable and may cause artifacts as the confinement coefficient is increased [5]. Suffering from the initial condition and the vorticity confinement itself, the regions with zero velocity become chaotic due to the fact that the vorticity force only depends on the particle vorticity. In addition, the large particle vorticity also causes artifacts analogous to the vorticity confinement.

In this paper, we propose an efficient adaptive vortex particle method that can be able to fix the problem in the original method and improve the performance. We explore our recently proposed spatial adaptive vorticity confinement [7], which varies the confinement strength with helicity to overcome the limitation of the vorticity confinement. We apply this method to the vortex particle method as the vorticity forcing. By integrating the spatial adaptive vorticity confinement, the infection of the vortex forcing is more rational. On the other hand, we also present an adaptive approach to handle particles efficiently. Since our vorticity forcing method eliminates the effect of the particles with zero velocity, making the kernel radius adapt with the particle velocity by following a hyperbolic function. Thus we greatly reduce the computational complexity of the traverse of the grid around particles. In addition, we have implemented the proposed methods using GPUs with CUDA to further accelerate the simulation. Finally, we procedurally synthesize the high resolution turbulence flows from the coarse simulation as what Yoon *et al.* [8] did. In summary, our approach has the following characteristics:

- A new vorticity forcing method that integrates the spatial adaptive vorticity confinement into the vortex particle method, which can overcome the problem of the original method and achieve more realistic smoke.
- An efficient adaptive approach is proposed to handle vortex particles. We adapt the particle influencing region with the velocity by a hyperbolic function and greatly reduce the computational complexity by ignoring the particles that are not contributed to the vorticity forcing.
- Highly turbulent effects like explosions can be generated easily with our model.
- Simulation and rendering are both performed on GPUs with CUDA, most of our algorithms are GPU-oriented. Turbulence effects can be simulated at interactive rates.

The rest of the paper is organized as follows: A brief overview of related work is presented in Section 2. The basic grid-based method is detailed in Section 3. Efficient adaptive vortex particle method is explained in Section 4. GPU implementation is presented in Section 5. Experimental results are given in Section 6. We conclude our work and give our future work in Section 7.

2. Related Work

Fluid simulation for computer graphics has been a major interesting area in the past few years. A recent book by Bridson [9] provides a relatively comprehensive review. We only review some recent studies here. Grid-based techniques [6], [10], [11] were proposed to provide interesting visual results, however, their developments are prohibited by the computational power available. On the other hand, procedural synthesis techniques [1], [2], [3] were developed to add details to these simulations with noise.

In addition, techniques for generating a higher resolution result from a lower resolution simulation are getting popular. Nielsen *et al.* [12] developed a method by using a low-resolution input simulation to guide the higher-resolution one where details are added. Yoon *et al.* [8] utilized the vortex particle method [5] to generate the vorticity field, which are then procedurally synthesized with the flow fields obtained from the incompressible Navier-Stokes solver to improve sub-grid visual details. Lentine *et al.* [13] presented a speedup technique for simulating detailed fluids by generating a divergence-free velocity field on a coarse grid. Zhao *et al.* [14] proposed a scheme utilizing random forcing in turbulence integration to enhance an existing fluid simulation with controllable turbulence. Chen *et al.* [15] developed a new Lagrangian primitive to incorporate turbulent flow details. In our recent work [7] a spatial adaptive vorticity confinement based on the helicity to achieve the visual pleasing effects of highly turbulent flows is proposed. The work presented in this paper is the application of the proposed spatial adaptive vorticity confinement to the vortex particle method.

Our focus in this paper is to achieve highly turbulent flows while keeping the performance that is suitable for real-time applications such as games. In this paper, we will utility the spatial adaptive vorticity confinement we proposed recently [7] to develop a new approach focusing on germinating highly turbulent flows in real time.

3. Grid-Based Simulation

In this section, we briefly review the basic incompressible Navier-Stokes equations first. Then the spatial adaptive vorticity confinement [7] will be reviewed along with the original vorticity confinement.

3.1. Incompressible Navier-Stokes Equations

The behavior of incompressible fluid is described by the Navier-Stokes (N-S) equations:

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \mu\nabla^2\mathbf{u} + \boldsymbol{f} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}$ denotes the velocity and p is the pressure, ρ is the mass density, and $\boldsymbol{f}$ represents the external forces such as gravity or vorticity confinement. The numerical methods for solving the incompressible N-S equations can be found in [9].

3.2. Spatial Adaptive Vorticity Confinement

Vorticity confinement was introduced to computer graphics by Fedkiw *et al.* [6]. This method re-injects the lost energy dissipation back to the flow. At the same time, a large coefficient would cause artifacts and instabilities. Spatial adaptive vorticity confinement [7] has been introduced by us recently to address this problem. The vorticity $\boldsymbol{\omega}$ can be obtained by computing the curl of velocity field $\mathbf{u}$:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} \tag{3}$$

The gradient of $|\boldsymbol{\omega}|$ is normalized to obtain the normalized vorticity location vector $\mathbf{N}$:

$$\mathbf{N} = \frac{\nabla|\boldsymbol{\omega}|}{|\nabla|\boldsymbol{\omega}||} \tag{4}$$

Vorticity confinement force is computed as follows [6]:

$$\boldsymbol{f}_{conf} = \epsilon h(\mathbf{N} \times \boldsymbol{\omega}) \tag{5}$$

where ϵ is the coefficient that controls the strength of the confinement by the user, and h is the mesh size.

In [7] we modified the original formulation where the strength of the confinement varies with respect to the helicity. Helicity is defined as $\mathbf{u} \cdot \boldsymbol{\omega}$. The confinement force can be obtained by factoring $|\boldsymbol{\omega}|$ out from Equation (5) and replacing ϵ with $\epsilon|\mathbf{u}|$, then it is converted to be dimensionless:

$$\boldsymbol{f}_{conf} = \epsilon_h h|\mathbf{u} \cdot \boldsymbol{\omega}|(\mathbf{N} \times \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}) \tag{6}$$

where ϵ_h is a true dimensionless parameter.

4. Efficient Adaptive Vortex Particle Method

Vortex particle method is a hybrid method that combines grid-based and particle-based methods to overcome the drawbacks in both methods. They are a class of Lagrangian numerical methods for the simulation of incompressible fluid flows where the vorticity is an integral component. The N-S

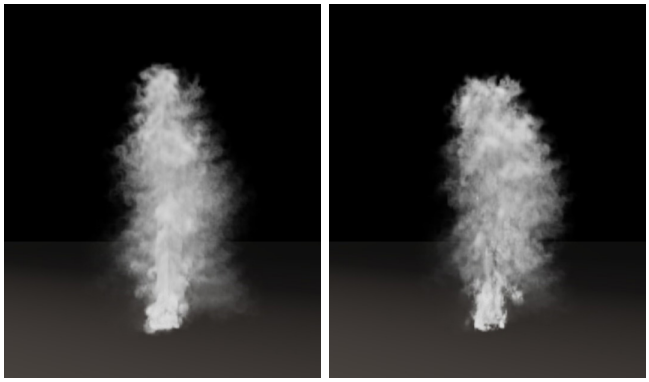

Figure 2. A comparison of using the spatial adaptive vorticity confinement (*Left*) and the vorticity confinement in [6] (*Right*) as the vorticity forcing method. We clamp the vorticity magnitude to a high range for both cases.

equations can be put into a vorticity form by taking the curl of Equation (1) to obtain:

$$\boldsymbol{\omega}_t + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} = \mu\nabla^2\boldsymbol{\omega} + \nabla \times \boldsymbol{f} \tag{7}$$

where $(\mathbf{u} \cdot \nabla)\boldsymbol{\omega}$ is the vorticity advecction term and $(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$ is the vortex stretching term. One of the outstanding advantages of this method is that it suffers no numerical dissipation of vorticity. However, in order to solve this equation, we need to reconstruct the velocity field from the particle vorticity to obtain the vorticity field, this is a rather complex process. Selle *et al.* [5] created a simple method to avoid this step entirely. They used the velocity field determined by the grid simulation instead of the particle vorticity. We implemented and improved their method in this paper. Each vortex particle has both a vector position and a vector strength associated with it. A kernel (Gaussian kernel) is used to define the influencing strength of the velocity field where the grid is around particles by the vorticity force. We ignore the $\mu\nabla^2\boldsymbol{\omega}$ and $\nabla \times \boldsymbol{f}$ terms the same as Selle *et al.* [5] did. After initializing the vorticity and location of vortex particles, the algorithm in a time step can be summarized as follows:

1: Advect particles
2: Compute the vortex stretching term $(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$
3: Influence the velocity field where the grid is around particles by the vorticity force

Particles are advected according to:

$$\mathbf{x}_p(t + \triangle t) = \mathbf{x}_p(t) + \triangle t\mathbf{u}_p \tag{8}$$

where $\mathbf{u}_p$ can be obtained by interpolating the background velocity with the particle position. The vorticity is stretched by:

$$\boldsymbol{\omega} \mathrel{+}= \triangle t(\boldsymbol{\omega}_x\frac{\partial\mathbf{u}}{\partial x} + \boldsymbol{\omega}_y\frac{\partial\mathbf{u}}{\partial y} + \boldsymbol{\omega}_z\frac{\partial\mathbf{u}}{\partial z}) \tag{9}$$

since this term is unstable, the vorticity magnitude could be exponentially increased, thus we clamped the magnitude of the vorticity.

4.1. Vorticity Forcing

A vortex particle is an influencing element, each individual element exerts its own influence throughout the flow field. Selle *et al.* [5] used the vorticity confinement to influence the flow field. Suffering from the initial condition (an initial vorticity magnitude is non-zero) and the vorticity confinement itself, the regions with zero velocity become chaotic due to the fact that the vorticity force only depends on the particle vorticity, and large particle vorticity may destroy the simulation analogous to the vorticity confinement. In order to address this problem, we integrate the spatial adaptive vorticity confinement [7] into the vorticity forcing method instead of the vorticity confinement to order to improve the visual effects of the simulation. A kernel function is used to decrease the confinement force strength with respect to the distance from the particle center. We use a clamped Gaussian kernel similar to the one used by Selle *et al.* [5]:

$$\xi_p(\mathbf{x}-\mathbf{x}_p) = \begin{cases} \frac{e^{-|\mathbf{x}-\mathbf{x}_p|^2/2r^2}}{(r^3(2\pi)^{3/2})} & 0 \le |\mathbf{x}-\mathbf{x}_p| \le r \\ 0 & otherwise \end{cases} \quad (10)$$

where r is the kernel radius. The vorticity within particle regions is defined as $\tilde{\boldsymbol{\omega}}_p = \xi_p(\mathbf{x}-\mathbf{x}_p)\boldsymbol{\omega}_p$. The direction from the particle center is $N_p(\mathbf{x}) = (\mathbf{x}_p-\mathbf{x})/|\mathbf{x}_p-\mathbf{x}|$. By applying the spatial adaptive vorticity confinement, the influencing function can be obtained:

$$\boldsymbol{F}_p = \epsilon_p|\mathbf{u}\cdot\tilde{\boldsymbol{\omega}}_p|(\mathbf{N}_p \times \frac{\tilde{\boldsymbol{\omega}}_p}{|\tilde{\boldsymbol{\omega}}_p|}) \quad (11)$$

where ϵ_p is defined as $|\boldsymbol{\omega}_p|$, it is the magnitude of the particle vorticity.

4.2. Efficient Adaptive Handling of Particles

The kernel radius determines how many grid cells are influenced by the vortex particles. The larger radius and more particles will greatly increase the computational cost, especially on GPUs. Suffering from GPU architecture and compilation rules, "for" loops with logic computation within a thread is not GPU friendly. Parallel computing is the strength of GPUs, reducing sequential computational complexity within a thread is the main point to improve the performance on GPUs. Hence we present an efficient adaptive method to handle particles. In our implementation, each particle is assigned to one thread, each thread has to traverse a three-dimensional (3D) cube. The size of a 3D cube depends on the kernel radius.

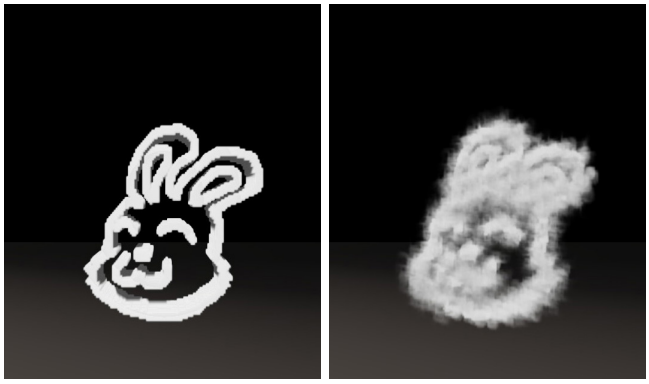

Figure 3. The smoke "rabbit" stays in the air without any velocity applied. The spatial adaptive vorticity confinement was used on the left case, and the original method was used on the right case.

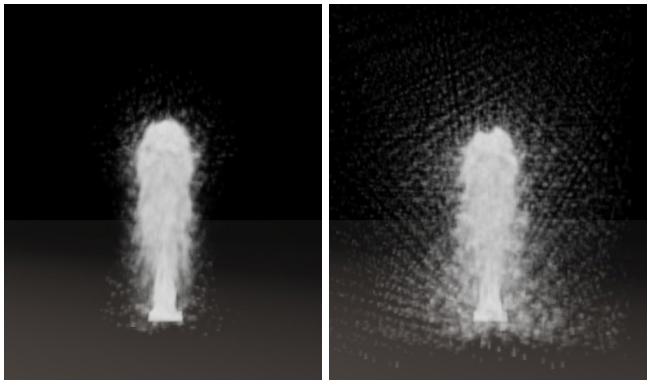

Figure 4. A comparison of using our efficient adaptive particle handling (*Left*) and without using it (*Right*). Each activated particle brings the density to show its state. Buoyancy force was ignored in this simulation.

In order to decrease the number of traversals, we adapt the influencing range with velocity by a hyperbolic tangent function. Since a hyperbolic tangent function $\tanh(x)$ tends to 1 as x approaches infinity, and $\tanh(x)$ equals to 0 when x is 0. By applying this function, the radius will be zero when the magnitude of velocity is zero, the influencing range of particle will be reduced with respect to the magnitude of velocity. Because the particle outside the smoke will be ignored and the kernel radius is adaptive, this approach will greatly reduce the computational cost, saving a significant amount of computation. Furthermore, the particle with low velocity decreases the influencing radius so that the smoke with low velocity will be more gathered. The influencing radius can be obtained by:

$$r = r_{max}\tanh(\sigma|\mathbf{u}_p|) = r_{max}\frac{e^{\sigma|\mathbf{u}_p|}-e^{-\sigma|\mathbf{u}_p|}}{e^{\sigma|\mathbf{u}_p|}+e^{-\sigma|\mathbf{u}_p|}} \quad (12)$$

where r_{max} is the maximum of influencing range, coefficient

σ determines the changing rate of the influencing range.

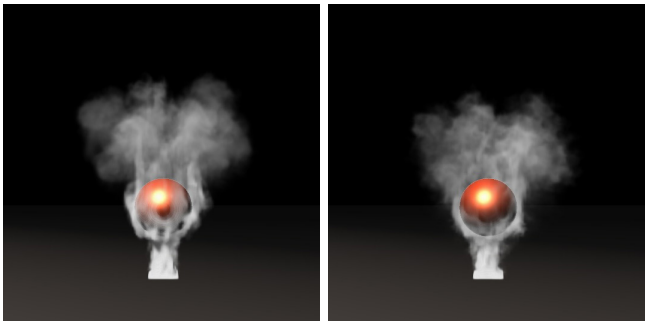

Figure 5. Smoke rising around a sphere shows the difference of using our efficient adaptive particle handling (*Left*) and without using it (*Right*). This example is a $128 \times 128 \times 128$ simulation with a $64 \times 64 \times 64$ coarse grid.

4.3. Turbulence Details Synthesis

To achieve a high resolution fluid simulation, up-sampling from a coarse simulation with procedurally synthesized turbulent details is an effective approach. This kind of techniques solve the N-S equations only on a coarse grid, avoiding the huge computational time by achieving a high resolution grid through up-sampling, leading to great performance improvement. Since the vortex particle method is independent from grid resolution, thus this method suits to generate high resolution turbulent details and no additional computation is needed. Our up-sampling method is similar to that of Yoon *et al.* [8]. But the difference is that the high resolution turbulence is synthesized from a coarse simulation using our efficient adaptive vortex particle method, the particle stretch, and the advection by a coarse velocity field. In order to retrieve more details during the up-sampling process, we use the cubic B-spline interpolation [16] instead of the trilinear interpolation. The flow chart of our proposed approaches to simulate and visualize smoke is illustrated in Figure 6.

5. GPU Implementation

Taking the advantages of parallel computing with CUDA, manifest performance improvement can be obtained. Our simulator is optimized for GPU implementation so that both grid-based and particle-based simulations can be processed in parallel. For grid-based simulation, an example thread assignment of using $32 \times 32 \times 32$ grid is shown in Figure 7. In this example, $16 \times 16 \times 1$ threads are assigned for a block. Each thread processes 16 grids along the z axis. Since the resource of the register within a block is limited, assigning too many threads within a block will reduce the

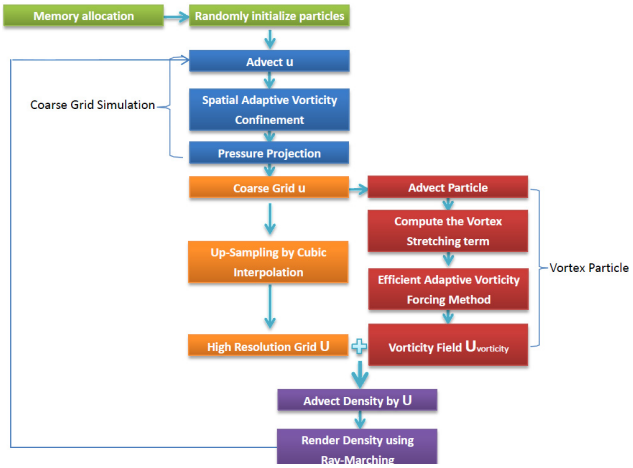

Figure 6. The flow chart of our proposed approaches. **u** and **U** are the velocity field in a coarse grid and a high resolution grid, respectively. $\mathbf{U}_{voritcity}$ is the vorticity field.

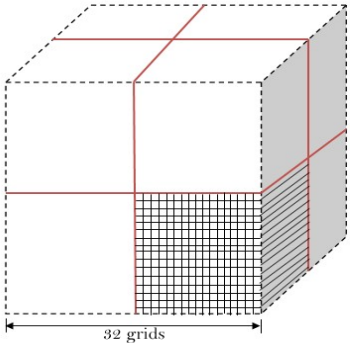

Figure 7. An example thread assignment using a $32 \times 32 \times 32$ grid in the grid-based simulation. The red line and black line separate the block and the thread, respectively. Within a block, $16 \times 16 \times 1$ threads are assigned. Each thread processes 16 grids along the z axis.

performance. In our experience, having 256 threads within a block is a quite well solution for our program. Since interpolation is commonly used in fluid simulation and using texture memory is a handy way, we use 3D texture memory to store the data, then interpolation can be done directly by positioning the input. During the procedural synthesis process, we do not need to allocate texture memory for high resolution velocity fields thanks to the coarse velocity field. Thus we save lots of computational cost since updating the large size texture memory will significantly reduce the performance. For particle-based simulation, we process each particle using a thread, so each particle can be easily corresponded by a thread. Particles are

controlled by three parameters: the vorticity, the location, and the influencing radius. Particles are seeded with a random position, a direction, and a vorticity magnitude. A periodic boundary condition for particles flow is used.

In addition to the solver, a volume renderer has also been implemented using GPU with CUDA. CUDA allows users to cooperate with the graphics APIs through binding the vertex buffer objects (VBOs) and the pixel buffer objects (PBOs). The results from the volume renderer can be directly rendered on the screen through PBOs, thus avoiding the transfer of the date back to CPU across the PCI-express bus.

6. Experimental Results

All results were performed on a PC with Intel Core i7 CPU, 6GB RAM, and NVIDIA GeForce GTX 480 graphics card. Solver and renderer were implemented with CUDA working on GPUs entirely. MacCormack method [17] was used as the advection approach, Jacobi iteration [18] for solving poisson equation. For balancing the visual effects and the performance, ray-marching [19] was used in our implementation.

Figure 1 shows a sequence of figures capturing the smoke rising around a sphere. This simulation was done on a $256 \times 256 \times 256$ grid up-sampled using a $128 \times 128 \times 128$ coarse velocity field at 8.2 frames per second (fps). It demonstrates that our model can generate realistic turbulent smoke effects. Figure 8 also shows a sequence of figures capturing the rising of the smoke "University badge" simulated with the same grid size.

A comparison between our vorticity forcing method and the original method are shown in Figures 2 and 3. Figure 2 demonstrates our vorticity forcing method did not cause artifacts but the vorticity confinement did while the vorticity magnitude was clamped in a high range. Figure 2 shows the smoke in zero velocity regions. Using the original method, the smoke became chaotic due to the fact that the vorticity force only depends on the vorticity. The figure on the left shows our forcing method with helicity considered did not affect the smoke within zero velocity regions.

Figures 4 and 5 compare the effects of using and without using our efficient adaptive particle handling method. In Figure 4, each activated particle brings density showing how our approach eliminates the particles that are useless. Figure 5 illustrates that how our approach make the smoke with low velocity be more gathered due to the influencing range reduced.

Method	Number of Particles	Influencing Raidus	Time (fps) (Incl. Rendering)
With E.A.V.P.	8192	4	41.9
Without E.A.V.P.	8192	4	33.5
With E.A.V.P.	16384	4	36.4
Without E.A.V.P.	16384	4	25.8
With E.A.V.P.	8192	8	25.4
Without E.A.V.P.	8192	8	13.2
With E.A.V.P.	16384	8	16.5
Without E.A.V.P.	16384	8	6.7

Table 1. A performance comparison of using and without using our efficient adaptive vortex particle (E.A.V.P.) method. All results were simulated on a $128 \times 128 \times 128$ grid up-sampled from a $64 \times 64 \times 64$ coarse grid.

Table 1 shows that our efficient adaptive vorticity confinement can improve the performance between 8 to 12 fps compared to the original method. When more particles and larger radius are set, more than two times speedups were achieved.

Our model can simulate turbulent smoke phenomena in real-time. Table 2 provides the performance information of the comparison of different turbulence models using procedural synthesis. By taking the parallel advantage of GPUs, our method is much faster than Yoon *et al.* [8]'s model where a CPU was used. In our recent work [7] we used the same graphics card as the one used in this paper and our model proposed here can get the slightly better performance.

7. Conclusion and Future Work

In this paper, we have proposed an efficient adaptive approach for producing highly turbulent effects in real-time. The spatial adaptive vorticity confinement is introduced to the vortex particle method. More rational and stable effects can be achieved by this method compared to the use of the original vorticity confinement. Considering that most of particles are useless during the simulation process, removing them can greatly reduce the performance. As a result, we have presented an efficient adaptive particle handling method to adapt the influencing range with respect to the velocity magnitude by a hyperbolic function, then those particles with zero velocity will be eliminated and the radius of low velocity particle will be reduced. Experimental results show that our efficient approach can greatly improve the performance compared to the original vorticity confinement. In addition, our model is faster compared to other procedural synthesis models [8], [7].

Multi-grid method is fast for solving the poisson equation. In our future work, we will try to use multi-grid method in

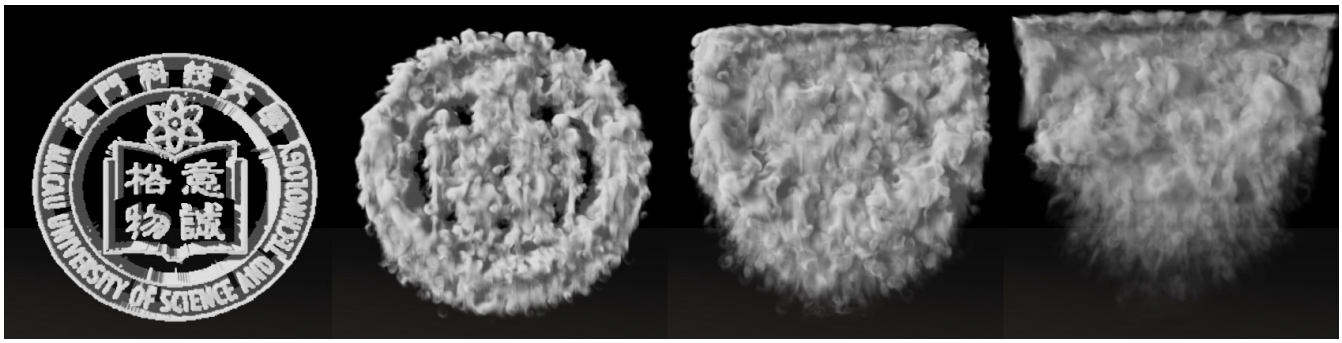

Figure 8. A set of sequential figures showing the rising of the smoke "university badge". The simulation was done on a $256 \times 256 \times 256$ grid up-sampled from a $128 \times 128 \times 128$ coarse grid.

Turbulence Model	GPU/CPU	Coarse Resolution	Target Resolution	Time (fps) (Include Rendering)
Our model	GPU	$64 \times 64 \times 64$	$128 \times 128 \times 128$	41.9
He *et al.* [7]	GPU	$64 \times 64 \times 64$	$128 \times 128 \times 128$	39.3
Our model	GPU	$128 \times 128 \times 128$	$256 \times 256 \times 256$	8.2
He *et al.* [7]	GPU	$128 \times 128 \times 128$	$256 \times 256 \times 256$	6.6
Yoon *et al.* [8]	CPU	$30 \times 90 \times 30$	$120 \times 360 \times 120$	0.11

Table 2. A performance comparison of our model with other turbulence models. All of these models use procedural synthesis. He *et al.* [7] and our model were run on a NVIDIA GTX480 GPU, while Yoon *et al.* [8]'s cases were performed on an Intel Quad Core CPU.

our simulator to further improve the performance. Moreover, obtaining more realistic and interesting details is also one of our goals. For example, Ma *et al.* [20] proposed a motion texture synthesis method to create interesting visual effects, exploring this technique may lead to some interesting results.

Acknowledgment

This work is supported by the National High-Technology Research and Development Program of China (2010AA122205). Special thanks to anonymous reviewers for their constructive comments on the paper.

References

[1] H. Schechter and R. Bridson. Evolving sub-grid turbulence for smoke animation. pages 1–8. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2008*, 2008.

[2] T. Kim, N. Thürey, D. James, and M. Gross. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics (SIGGRAPH Proc.)*, 27:Article 50, 2008.

[3] R. Narain, J. Sewall, M. Carlson, and M. C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics (SIGGRAPH Asia Proc.)*, 27:Article 166, 2008.

[4] M. N. Gamito, P. F. Lopes, and M. R. Gomes. Two dimensional simulation of gaseous phenomena using vortex particles. pages 3–15. In *Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*, 1995.

[5] A. Selle and R. Rasmussen, N.and Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (SIGGRAPH Proc.)*, 24:910–914, 2005.

[6] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. pages 15–22. In *Proceedings of ACM SIGGRAPH 2001*, 2001.

[7] S. He, H. C. Wong, W. M. Pang, and U. H. Wong. Real-time smoke simulation with improved turbulence by spatial adaptive vorticity confinement. *Computer Animation and Virtual Worlds (CASA2011 Special Issue)*, 22:107–114, 2011.

[8] J.-C. Yoon, H. R. Kam, J.-M. Hong, S. J. Kang, and C.-H. Kim. Procedural synethesis using vortex particle method for fluid simulation. *Computer Graphics Forum (Pacific Graphics Proc.)*, 28:1853–1859, 2009.

[9] R. Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[10] J. Stam. Stable fluids. pages 121–128. In *Proceedings of ACM SIGGRAPH 1999*, 1999.

[11] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. pages 181–188. In *Proceedings of ACM SIGGRAPH 1997*, 1997.

[12] N. B. Nielsen, B. B. Christensen, N. B. Zafar, D. Roble, and K. Museth. Guiding of smoke animations through variational coupling of simulations at different resolution. pages 217–226. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2009*, 2009.

[13] M. Lentine, W. Zheng, and R. Fedkiw. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Transactions on Graphics (SIGGRAPH Proc.)*, 29:Article 114, 2010.

[14] Y. Zhao, Z. Yuan, and F. Chen. Enhancing fluid animation with adaptive, controllable and intermittent turbulence. pages 75–84. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium of Computer Animation 2010*, 2010.

[15] F. Chen, Y. Zhao, and Z. Yuan. Langevin particle: A self-adaptive lagrangian primitive for flow simulation enhancement. *Computer Graphics Forum (EUROGRAPHICS Proc.)*, 30:435–444, 2011.

[16] C. Sigg and M. Hadwiger. Fast third-order texture filtering. pages 313–329. In *GPU Gems 2*, 2005.

[17] A. Selle, R. Fedkiw, B. Kim, L. Liu, and J. Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35:350–371, 2008.

[18] K. Crane, S. Tariq, and I. Llamas. Real time simulation and rendering of 3d fluids. pages 633–675. In *GPU Gems 3*, 2008.

[19] J. M. Cohen, S. Tariq, and S. Green. Interactive fluid-particle simulation using translating eulerian grids. pages 15–22. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2010*, 2010.

[20] C. Ma, L. Wei, B. Guo, and K. Zhou. Motion field texture synthesis. *ACM Transactions on Graphics (SIGGRAPH Asia Proc.)*, 28:Article 110, 2009.