A Data-driven Approach for Synthesizing High-resolution Animation of Fire

Syuhei Sato*
Hokkaido University

Takuya Morita†
Hokkaido University

Yoshinori Dobashi‡
Hokkaido University
CREST, JST

Tsuyoshi Yamamoto§
Hokkaido University

Abstract

We propose a simple and efficient data-driven method for synthesizing high-resolution 3D animations of fire from low-resolution fluid simulations. Our method is based on grid-based fluid simulation. The key concept behind our method is to use a precomputed database of high-resolution velocity fields in order to produce small-scale details that are lost in low-resolution velocity fields. The database is constructed by 2D fluid simulation and no high-resolution 3D fluid simulations need to be executed. At run-time, a low-resolution 3D fluid simulation is executed and the velocity field calculated at each time step is approximated by a linear combination of the precomputed velocity fields. This approximation process produces realistic small-scale detail. Using our method, users can efficiently design animations of fire with low-resolution simulation and our method converts them into high-resolution animations. We examine the ability of our method by applying it to simulations of fire under various situations including moving obstacles.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.6 [Computer Graphics]: Methodology and Techniques;

Keywords: fluid simulation, fire, upsampling

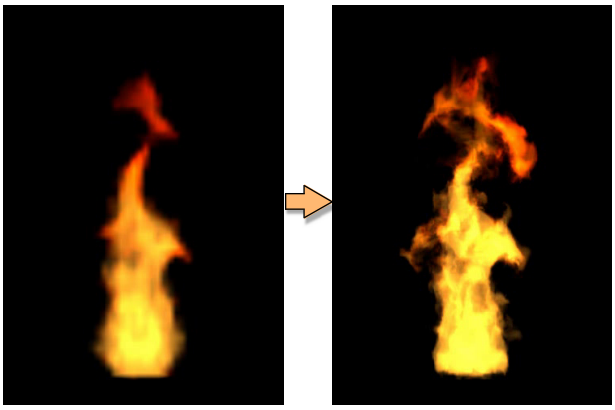

Figure 1: *Simulated low-resolution fire (left,* $32 \times 32 \times 64$ *grid points) and a high-resolution fire synthesized using our method (right,* $128 \times 128 \times 256$ *grid points). The computation time of the simulation (left) was 0.05 sec for each time step and the computation time for the synthesis of the fire on the right took an additional 0.34 sec.*

1 Introduction

Visual simulation of fluids has become one of the most important research topics in computer graphics. Many methods have been proposed for simulating smoke, water, fire, and so on [Bridson 2008]. Most of the recent methods are based on computational fluid dynamics to create realistic animations and these are used in many applications such as movies and computer games. However, one of the problems in simulating fluids using computational fluid dynamics is the expensive computational cost. In entertainment applications such as movies, creating the motion of fluids is often requested. Animators usually try to create the desired motion by repeating fluid simulations with different parameter settings until a satisfactory result is obtained. However, this is an extremely tedious and time-consuming task incurring an expensive computational cost.

Many methods have been proposed to address the problem mentioned above. A straightforward approach is to accelerate the computation [Crane et al. 2007; Lentine et al. 2010]. This approach would be the ultimate solution if real-time performance could be achieved for fluid simulations of arbitrary resolution. However, this has not yet been achieved. Another approach is to control the fluid simulation to create the desired motion [Treuille et al. 2003; Fattal and Lischinski 2004]. This approach is also promising but a trial-and-error process is still required to tune the control parameters. Several recent research projects have focused on a different approach: using low-resolution fluid simulation as a guide to produce high-resolution results. In this approach, animators design the desired animation of the fluid efficiently using low-resolution fluid simulation. The low-resolution animations are subsequently converted into high-resolution animations. Some methods control the high-resolution simulation using low-resolution simulation results [Nielsen and Christensen 2010; Yuan et al. 2011]. However, in these methods, costly high-resolution simulations need to be executed to verify the quality of the final result. Methods combining noise functions with fluid simulation [Kim et al. 2008; Schechter and Bridson 2008; Pfaff et al. 2010], on the other hand, can produce detailed animations without running high-resolution simulations. However, results obtained by these methods are less realistic than those obtained by physical simulations. Our method also produces high-resolution results without running the high-resolution simulations but uses precomputed velocity fields obtained by fluid simulation, resulting in enhanced realism compared to noise-based methods.

Our method is based on grid-based fluid simulation and is suitable for synthesizing animations of gaseous phenomena such as fire. Our motivation comes from the observation that we often see similar flow patterns at different times and positions on different scales during animations of gaseous objects. This may be the reason why the procedural approach, such as fractals, can produce realistic-looking animations. Our method generates such flow patterns by 2D fluid simulation and uses them to synthesize a high-resolution velocity fields from low-resolution velocity fields. The method consists of two processes. First, in a preprocessing step, a

*e-mail: sato@ime.ist.hokudai.ac.jp
†e-mail:morita@ime.ist.hokudai.ac.jp
‡e-mail:doba@ime.ist.hokudai.ac.jp
§e-mail:yamamoto@ist.hokudai.ac.jp

database of high-resolution velocity fields is constructed by running a fluid simulation. An important feature of the method is that the precomputed database is constructed by *2D fluid simulation*. Next, at run-time, a high-resolution velocity field is synthesized from the low-resolution velocity field by approximating the low-resolution velocity field with a linear combination of the precomputed velocity fields. The approximation is done in such a way that the downsampled version of the resulting high-resolution velocity field becomes identical to the low-resolution velocity field.

Fig. 1 shows an example of a synthetic fire created using our method. The image on the left shows the fire obtained from a low-resolution simulation with $32 \times 32 \times 64$ grid points. The high-resolution fire on the right is created by converting the low-resolution fire to a four times higher resolution, that is, $128 \times 128 \times 256$ grid points. The low-resolution simulation takes 0.05 sec for each time step and our method requires only an additional 0.34 sec to produce the high-resolution results.

The method has three features:

- The database is created by 2D fluid simulation. We use 2D velocity fields to add small-scale detail to the 3D low-resolution velocity field. This results in a significant reduction in computational costs for both precomputation and the run-time process.
- Using our synthesis method recursively, animations can ideally be synthesized with arbitrarily high resolution.
- The method is highly suitable for parallel computation. The low-resolution velocity field is subdivided into small blocks and the high-resolution velocity field can be synthesized in parallel for each block.

The rest of this paper is organized in the following way. In Section 2, we briefly discuss some related work to clarify the advantages of our method. Next, in Section 3, the fluid solver used in this paper is briefly described. Section 4 describes the details of our method. Some experimental results are shown in Section 5. Section 6 discusses the limitations of our method. Finally, in Section 7, we give our concludes to this paper.

2 Related Work

Stam [1999] addressed the stability problem in solving the Navier-Stokes equations and made fluid simulation practical. Following this work, many methods were proposed for simulating various fluid phenomena. Readers can find details of those methods in [Bridson 2008]. One of the problems with fluid simulation, however, is the expensive computational cost. Therefore, many methods that reduce the cost have been proposed [Losasso et al. 2004; Feldman et al. 2005; Crane et al. 2007; Batty et al. 2007; Dobashi et al. 2008; Lentine et al. 2010]. However, high-resolution simulation is still costly and tuning simulation parameters with repeated simulations is time-consuming. Some researchers use 2D fluid simulations for efficiently creating high-resolution animations of explosions [Rasmussen et al. 2003] or fire [Horvath and Geiger 2009]. These methods can reduce the computational cost significantly compared to high-resolution 3D fluid simulations. However, they are still time-consuming since they need to simulate high-resolution 2D fluids multiple times. In contrast to these methods, our method precomputes a single set of high-resolution velocity fields by running a 2D fluid simulation only once and synthesizes the high-resolution 3D velocity fields from the low-resolution 3D fluid simulation.

Recently, several methods have been developed to create high-resolution results from low-resolution simulation. Nielsen et al. [Nielsen et al. 2009; Nielsen and Christensen 2010] proposed methods for simulating high-resolution fluids that resemble a reference low-resolution simulation result. Yuan et al. [2011] proposed a method that regulates high-resolution fluid simulation with flow patterns extracted from low-resolution fluid simulations. Although these methods can generate high-resolution results with the desired fluid behavior, costly high-resolution simulations are required to produce the final animations. By combining low-resolution fluid simulation with turbulent noise functions, high-resolution results can be synthesized without conducting any high-resolution fluid simulations [Kim et al. 2008; Schechter and Bridson 2008; Pfaff et al. 2010]. These methods can add small scale details but the results are not very realistic when compared to the results obtained by fluid simulations.

Our method lies somewhere between the noise-based approach and the high-resolution simulation approach. It uses high-resolution 2D simulation to create a database of velocity fields and uses these to procedurally generate high-resolution results from low-resolution simulations. Treuille et al. [2006] and Wicke et al. [2009] also proposed data-driven methods for accelerating fluid simulations. However, our purpose is to convert low-resolution simulations into high-resolution results, not to accelerate the simulation itself. Furthermore, their methods require extremely long precomputation times ranging from twenty to thirty hours since their database is constructed by 3D fluid simulations. In our method, the precomputation time is less than one hour since 2D fluid simulations are used to construct the database.

3 Fluid Solver

In this paper, for simulating motions of fire, we assume the fluid to be inviscid and incompressible. The motion of the gases can be calculated by solving the following Navier-Stokes equations.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}, \quad (1)$$

$$\nabla \cdot \mathbf{u} = \mathbf{0}, \quad (2)$$

where $\mathbf{u}$ is the velocity field of the fluid, ρ is the density of the fluid, p is the fluid pressure, and $\mathbf{f}$ represents any external force such as gravity or wind. We use the GPU-accelerated method [Crane et al. 2007] to solve the above equations numerically. We also use the vorticity confinement method [Fedkiw et al. 2001] in order to add detailed turbulent motion.

For the simulation of fire, we take into account the temperature and the buoyancy force. These are calculated by the method described in [Crane et al. 2007]. Our method is applied only to the velocity field $\mathbf{u}$. Other physical quantities for the high-resolution grid are obtained by advecting them along with the synthesized high-resolution velocity field.

4 The Method

Fig. 2 shows an overview of our method, which comprises two processes: the prerocess and the run-time process. In the preprocess, a high-resolution 2D velocity field database is constructed. Next, in the run-time process, a low-resolution 3D velocity field is obtained by 3D fluid simulation and converted to a high-resolution velocity field using the database. After that, the scalar quantities are advected according to the high-resolution velocity field. In order to clarify the explanation, we assume that 2D simulations are done in uv space and 3D simulations are in xyz space. Details of these processes are described in the following subsections.

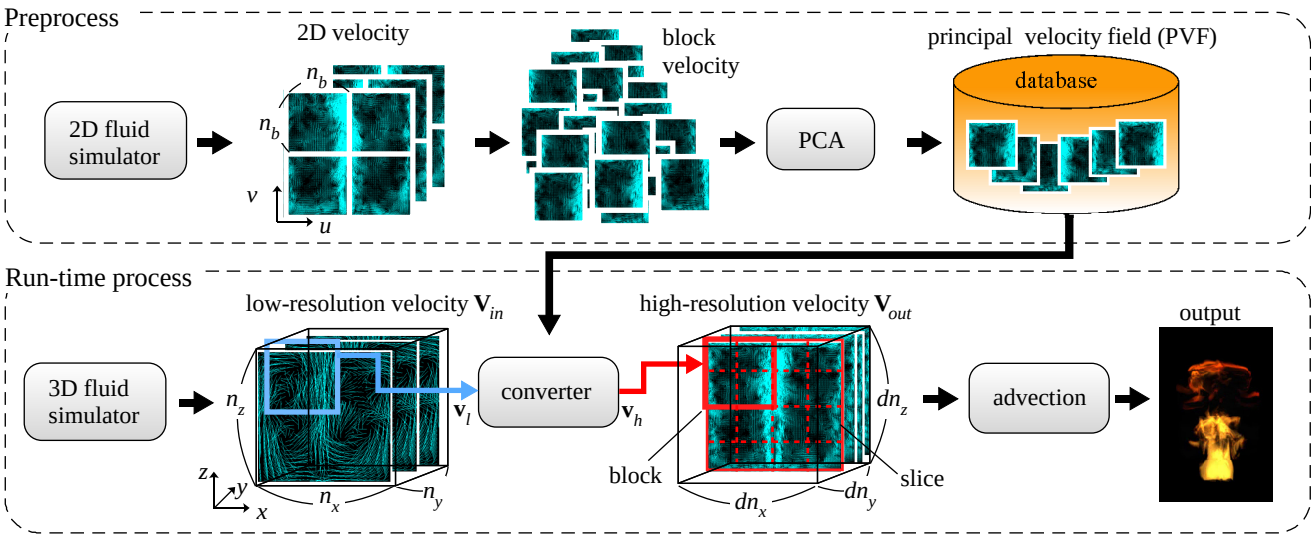

Figure 2: *Overview of our method.*

4.1 Database Construction

The database is constructed by running a 2D fluid simulation. During the simulation, our method temporarily stores the velocity field at each time step. After the fluid simulation has finished, each velocity field is subdivided into small blocks. We assume that each block is a square and the number of grid points of each block is $n_b \times n_b$. Then, our method applies principal component analysis and extracts the principal components of the block velocities. The number of principal components to be extracted is specified by the user but we found that thirty two components were sufficient for the examples shown in our paper. We call the principal components of the block velocity field *principal velocity fields*, or PVFs for short. The PVFs are stored in the database. In the following, the PVFs are denoted by $\mathbf{b}_i$ $(i = 0, \cdots, m-1)$, where m is the number of principal components.

The resolution of the 2D simulation should be the same as that of the high-resolution velocity field we want to generate. The simulation parameters should be chosen so that the desired turbulent details are captured. We cannot synthesize detailed turbulent motion in the subsequent 3D simulation if such turbulence is not included in the 2D simulation. However, we do not have to take into account any obstacles in the data construction process.

4.2 Synthesizing a High-resolution 3D Velocity Field

At run-time, a low-resolution 3D fluid simulation is executed. At each time step of the simulation, the 3D velocity field is converted into a high-resolution velocity field by our converter using the PVFs stored in the database. Let us denote the input 3D velocity field by $\mathbf{V}_{in}(n_x, n_y, n_z)$ where $n_x \times n_y \times n_z$ is the number of grid points. Our purpose is to create a 3D velocity field $\mathbf{V}_{out}(dn_x, dn_y, dn_z)$ where the resolution is d times higher. In our method, each slice of $\mathbf{V}_{out}$ is separately converted into a high-resolution velocity field. Let us consider the slice indicated by the large red rectangle in Fig. 2. This slice is further subdivided into small blocks indicated by the small red rectangle in the figure. The blocks overlap in order to reduce the discontinuity between them. In our current implementation, half of each block overlaps with its neighboring blocks. The high-resolution velocity field is created on a block basis and the velocities in the overlapping regions are linearly interpolated. Let us

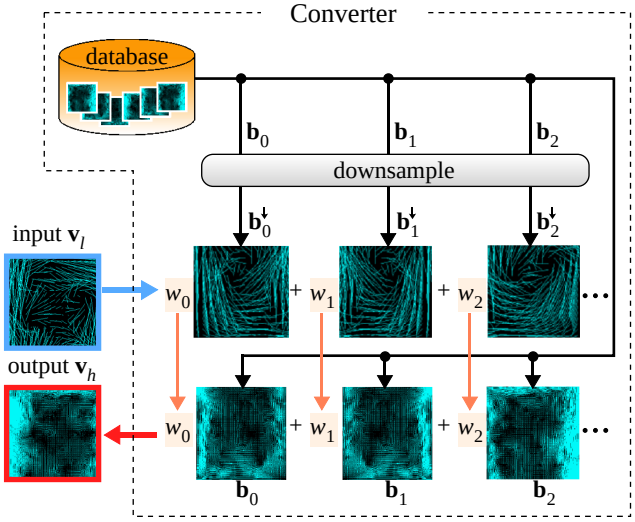

Figure 3: *Details of our converter.*

denote the velocity field in the red block by $\mathbf{v}_h$. A low-resolution velocity field $\mathbf{v}_l$ corresponding to $\mathbf{v}_h$ (indicated by the blue rectangle) is generated by linear interpolation of the input velocity field. Our converter synthesizes $\mathbf{v}_h$ from $\mathbf{v}_l$.

The details of the converter are illustrated in Fig. 3. First, our method downsamples PVFs to the resolution of the input velocity field; then, it approximates the input velocity $\mathbf{v}_l$ by a weighted sum of the downsampled PVFs. The approximation is done in a least squares manner as described in the next paragraph. The weights used for the approximation are then applied to the original PVFs. The output velocity $\mathbf{v}_h$ is calculated by the weighted sum of the PVFs. However, the dimensions of the input velocity field (3D) are different from those of the PVFs in the database (2D). Therefore, we apply the above process three times for each of the xyz components of the input velocity field. The horizontal components in 2D (i.e., u component of PVFs) are used for the horizontal components in the input 3D velocity fields (i.e., the x and y components), and the vertical components in 2D (v) are used for the vertical compo-

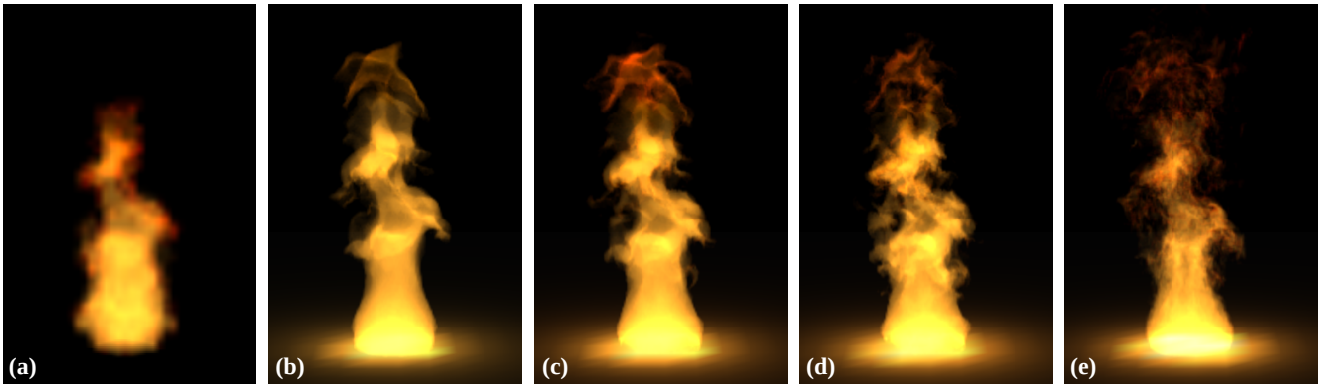

Figure 4: *Comparison of synthetic fire. (a) simulated low-resolution fire, (b) upsampled fire by linear interpolation, (c) high-resolution fire synthesized using our method, (d) high-resolution fire synthesized using our method with enhanced detailed motion, (e) high-resolution fire with wavelet turbulence.*

nents (z) in 3D.

Let us denote the u or v component of ith PVF by $\mathbf{b}_{i,*}$ ($* = u, v$). Similarly the x, y or z components of the low-resolution 3D velocity field are denoted by $\mathbf{v}_{l,\star}$ ($\star = x, y, z$). Our converter approximates $\mathbf{v}_{l,\star}$ with $\mathbf{b}_{i,*}$ by solving the following minimization problems.

$$\min_{w_{i,x}} \|\mathbf{v}_{l,x} - \sum_{i=0}^{m-1} w_{i,x}(\mathbf{b}_{i,x}^{\downarrow})\|^2, \tag{3}$$

$$\min_{w_{i,y}} \|\mathbf{v}_{l,y} - \sum_{i=0}^{m-1} w_{i,y}(\mathbf{b}_{i,x}^{\downarrow})\|^2, \tag{4}$$

$$\min_{w_{i,z}} \|\mathbf{v}_{l,z} - \sum_{i=0}^{m-1} w_{i,z}(\mathbf{b}_{i,y}^{\downarrow})\|^2, \tag{5}$$

where $\mathbf{b}_{i,*}^{\downarrow}$ indicates the downsampled PVF and m is the number of PVFs. Each of the above minimization problems results in a matrix equation. In the following, we refer only to the x component (i.e., Eq. 3). Other components are computed in a similar way.

Solving Eq. 3 results in the following matrix equation.

$$\mathbf{A}_x \mathbf{w}_x = \mathbf{c}_x, \tag{6}$$

where $\mathbf{A}_x$ is a $m \times m$ matrix, $\mathbf{w}_x$ and $\mathbf{c}_x$ are m dimensional column vectors. The elements of $\mathbf{w}_x$ are the weight $w_{i,x}$. (i, j) element of $\mathbf{A}_x$, $a_{ij,x}$, and ith element of $\mathbf{c}_x$, $c_{i,x}$, are given by:

$$a_{ij,x} = \mathbf{b}_{i,x}^{\downarrow} \cdot \mathbf{b}_{j,x}^{\downarrow} \tag{7}$$

$$c_{i,x} = \mathbf{v}_{l,x} \cdot \mathbf{b}_{i,x}^{\downarrow} \tag{8}$$

The weights can then be obtained by multiplying the inverse matrix of $\mathbf{A}_x$ with $\mathbf{c}_x$. After computing the weight, the high-resolution velocity field $\mathbf{v}_{h,x}$ for the block is synthesized using the following equation.

$$\mathbf{v}_{h,x} = \sum_{i=0}^{m-1} \alpha_i w_{i,x} \mathbf{b}_{i,x}. \tag{9}$$

In the above equation, we introduce a nonnegative control parameter α_i which is specified by the user at run-time. This parameter allows the user to adjust the degree of detailed motion to be added. Since we use principal component analysis in the database construction process, the small-scale turbulent motion tends to be represented by PVFs of a higher order. By increasing α_i for higher order PVFs, the user can enhance the detailed motion.

As shown in Eq. 7, the coefficient matrix $\mathbf{A}_x$ does not depend on the input velocity field and thus its inverse $\mathbf{A}_x^{-1}$ can be precomputed. Therefore, the computations involved in the synthesis of the high-resolution velocity field are a set of matrix and vector operations. We use GPU to efficiently calculate these operations. Furthermore, the computation of the high-resolution velocity field for each block is completely independent of other blocks. So, we further accelerate the computation by processing each block in parallel on the GPU.

4.3 Recursive Synthesis

The method proposed in the previous subsections fails when the resolution ratio is too high. This is because the high-dimensional space spanned by the principal block velocity fields becomes redundant when they are downsampled. This makes the coefficient matrix appearing in Eq. 6 singular. Therefore, there is a certain upper limit to the resolution that can be synthesized by our method. From our experiments, we found that the ratio should be less than four in most cases. In order to break through the upper limit, we apply our method recursively. That is, the input velocity $\mathbf{V}_i(n_x, n_y, n_z)$ is converted to $\mathbf{V}_1(dn_x, dn_y, dn_z)$ by our synthesizer. Next, $\mathbf{V}_1(dn_x, dn_y, dn_z)$ is converted again to the $\mathbf{V}_2(d^2n_x, d^2n_y, d^2n_z)$ velocity field. At each stage of recursion, we perform a 2D simulation of the desired resolution and reconstruct the database. By repeating this process, we can create arbitrarily high resolution for the velocity fields.

5 Results

We apply our method to simulating fire in the following way. After the high-resolution velocity field is synthesized, temperature of the fire is advected. We prepare a high-resolution grid for storing the temperature field. At each time step of the low-resolution 3D simulation, temperature of the source of the fire is added to the high-resolution grid and advected along with the high-resolution velocity field. In the following examples, the size of the principal block velocity field is 32×32 and the number of principal components is 32. Video files for the following examples can be found in the supplemental material.

First, we simulate fire in order to compare our method with other different methods. The database is constructed by the 2D simula-

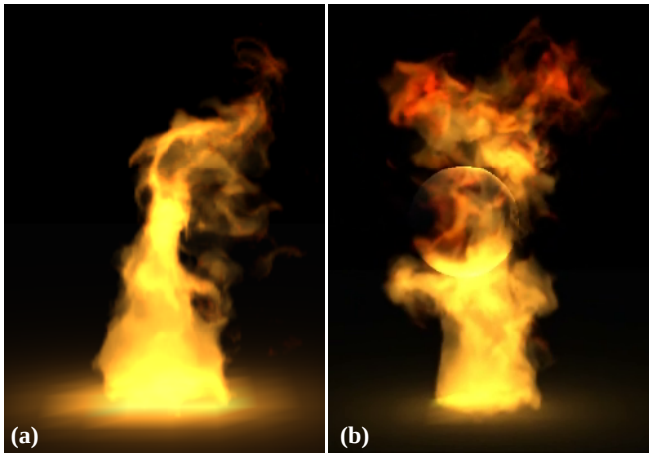

Figure 5: *Effects of wind forces (a) and moving obstacle (b).*

tion with 128×256 grid points. Figs. 4 (a) through (e) show a comparison of fire simulated using the different methods. Fig. 4(a) shows the low-resolution simulation with $32 \times 32 \times 64$ grid points. Fig. 4(b) shows fire on a high-resolution grid of $128 \times 128 \times 256$ created by upsampling the velocity field using linear interpolation. Figs. 4(c) and (d) show high-resolution results synthesized by our method from Fig. 4(a). In Fig. 4(d), the control parameter α_i is modified to enhance the small-scale turbulence. Fig. 4(e) shows the result using wavelet turbulence method [Kim et al. 2008]. The wavelet turbulence function successfully adds small-scale details but the result looks noisy. When compared to the result using the wavelet turbulence, our result seems to be more realistic.

Fig. 5 shows simulations of fire taking into account wind forces and moving obstacles. Although the wind and the obstacles are not taken into account in the database construction, our method successfully synthesizes realistic animations.

Finally, Fig. 6 is created using our method recursively. The high-resolution fire on a $256 \times 256 \times 512$ grid is synthesized by applying our method three times to a low-resolution fire simulated on a 32 grid.

The examples shown in this section are calculated on a PC with an Intel Core i7-2600K (CPU) and NVIDIA GeForce GTX 580 (GPU). Both the simulation and the rendering are executed on the GPU. For the precomputation, 2D fire is simulated on a 128×256 grid and it took 48 sec to create the database. The low-resolution 3D simulation on a $32 \times 32 \times 64$ grid took 0.05 sec for each time step. The synthesis of the high-resolution animation ($128 \times 128 \times 256$) took 0.34 sec. The time required for the full high-resolution simulation was 0.91 sec. Our method is three-times faster on average than the full high-resolution simulation.

6 Discussion

The high-resolution scalar field, such as for smoke density, sometimes becomes different to that of the low-resolution field. Although our method tries to match the high-resolution velocity field to the low-resolution velocity field, they are not completely the same. Therefore, this results in a difference in the distributions of the scalar quantities between high- and low-resolution. Currently, we are not considering compensating for this problem. Increasing/decreasing the scalar values artificially to reduce the difference would be required.

Figure 6: *Example of recursive synthesis. This example is created by applying our method three times to the low-resolution result ($32 \times 32 \times 64$) to synthesize fire on $256 \times 256 \times 512$ grid.*

Since our method creates a high-resolution velocity field for each block independently, the minimum memory requirement is proportional to the number of grid points of the block. The memory requirement for the precomputed database is very small since only 2D velocity fields are stored. However we need to prepare a set of high-resolution grids to store the final velocity field and the advected scalar fields. If we use particles to track the scalar quantities, and store the velocities at the particle positions, we do not have to prepare these high-resolution grids, as indicated by [Kim et al. 2008].

7 Conclusion

We have proposed a method for synthesizing high-resolution gaseous animations using low-resolution fluid simulations. Our method uses a precomputed database of the 2D velocity fields. The database can be constructed efficiently since a 2D simulation is executed only once. Our method has succeeded in creating high-resolution animations of fire. We are now planning to extend our method to the simulation of other phenomena, such as smoke, clouds, and liquids.

References

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transaction on Graphics 26*, 3, 100.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. AK Peters.

CRANE, K., LLAMAS, I., AND TARIQ, S. 2007. *Real Time Simulation and Rendering of 3D Fluids*. Addison-Wesley, ch. 30.

DOBASHI, Y., MATSUDA, Y., YAMAMOTO, T., AND NISHITA, T. 2008. A fast simulation method using overlapping grids for interactions between smoke and rigid objects. *Computer Graphics Forum 23*, 3, 539–546.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Transactions on Graphics 23*, 3, 439–446.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proc. SIGGRAPH 2001*, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Transaction on Graphics 24*, 3, 904–909.

HORVATH, C., AND GEIGER, W. 2009. Directable, high-resolution simulation of fire on the gpu. *ACM Transaction on Graphics 28*, 3, Article 41.

KIM, T., THÜREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. *ACM Transaction on Graphics 27*, 3, Article 50.

LENTINE, M., ZHENG, W., AND FEDKIW, R. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Transaction on Graphics 29*, 4, Article 114.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transaction on Graphics 23*, 3, 457–462.

NIELSEN, M. B., AND CHRISTENSEN, B. B. 2010. Improved variational guiding of smoke animations. *Computer Graphics Forum 29*, 2, 705–712.

NIELSEN, M. B., CHRISTENSEN, B. B., ZAFAR, N. B., ROBLE, D., AND MUSETH, K. 2009. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2009*, 217–226.

PFAFF, T., THÜREY, N., COHEN, J., TARIQ, S., AND GROSS, M. 2010. Scalable fluid simulation using anisotropic turbulence particles. *ACM Transaction on Graphics 29*, 6, Article 174.

RASMUSSEN, N., NGUYEN, D. Q., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. *ACM Transactions on Graphics 22*, 3 (aug), 703–707.

SCHECHTER, H., AND BRIDSON, R. 2008. Evolving sub-grid turbulence for smoke animation. In *Proc. the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–7.

STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH 1999, Annual Conference Series*, 121–128.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics 22*, 3 (Jul), 716–723.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transaction on Graphics 25*, 3, 826–834.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Transaction on Graphics 28*, 3, Article 39.

YUAN, Z., CHEN, F., AND ZHAO, Y. 2011. Pattern-guided smoke animation with lagrangian coherent structure. *ACM Transaction on Graphics 30*, 6, Article 136.