View-Dependent Multiscale Fluid Simulation

Yue Gao [1], Chen-Feng Li [2], Bo Ren [1] and Shi-Min Hu [1]

Technical Report **TR-110830**

Tsinghua University, Beijing, China

[1] Department of Computer Science and Technology, Tsinghua University

[2] College of Engineering, Swansea University

View-Dependent Multiscale Fluid Simulation

Yue Gao, Chen-Feng Li, Bo Ren, and Shi-Min Hu

Abstract—Fluid motions are highly nonlinear and non-stationary, with turbulence occurring and developing at different length and time scales. In real-life observations, the multiscale flow generates different visual impacts depending on the distance to the viewer. We propose a new fluid simulation framework that adaptively allocates computational resources according to the human visual perception. First, a 3D empirical model decomposition scheme is developed to obtain the velocity spectrum of the turbulent flow. Then, depending on the distance to the viewer, the fluid domain is divided into a sequence of nested simulation partitions. Finally, the multiscale fluid motions revealed in the velocity spectrum are distributed non-uniformly to these view-dependent partitions, and the mixed velocity fields defined on different partitions are solved separately using different grid sizes and time steps. The fluid flow is solved at different spatial-temporal resolutions, such that higher-frequency motions closer to the viewer are solved at higher resolutions and vice versa. The new simulator better utilizes the computing power, producing visually plausible results with realistic fine-scale details in a more efficient way. It is particularly suitable for large scenes with the viewer inside the fluid domain. Also, as high-frequency fluid motions are distinguished from low-frequency motions in the simulation, the numerical dissipation is effectively reduced.

Index Terms—fluid simulation, Hilbert-Huang transform, fluid velocity spectrum, view-dependent partition.

1 Introduction

Fluid simulations based on the Navier-Stokes equations have achieved great success in computer graphics. Many compelling methods with impressive animations have been reported in the past decade. However, fluid simulation remains a challenging task where improving the visual effect of fine-scale fluid motions and reducing the demand of computational resources are the main concerns. Unlike computational physics, the focus of graphics applications is on the visual effect of the final rendered images and animations. This implies a high potential value for exploiting the unique viewing information to improve existing fluid simulators. In this work, we propose a novel approach which incorporates the viewing information into the fluid solver and adaptively simulates the fluid at multiple scales, such that the computational resources are allocated to the key regions and to the key scales that have important impacts on the visual impression of turbulent fluids. This approach is particularly suitable for large scenes with the viewer immersed in the fluid domain. Such kind of scenes are rare to be seen in previous publications, but are often most desired by movie directors and game designers.

Generally, the techniques considering the viewer are referred as the levels of details, which has become a standard tool widely used in 3D geometry representation and texture rendering. The basic idea is that when the object is far from the viewer, a reduced geometry representation or a reduced texture is applied. This simplification is supported by the fact that, for human visual perception, the higher frequency signals play a more important role when the viewer is nearby, while the lower frequency signals are more important when the viewer is at distance [1].

Inspired by view-dependent rendering techniques, we first decompose the fluid velocity field into a series of frequency components using a modified empirical mode decomposition (EMD) method. Higher frequency components represent smaller scale fluid motions (typically local turbulent flow), while lower frequency components represent motions at larger scales (typically large eddies and global laminar flow). Also, the fluid domain is divided into a series of nested partitions centered with respect to the viewing frustum. Different grid sizes and time steps are assigned to different partitions depending on their distances to the viewer. The control of levels of details is then applied to each frequency component by distributing it non-uniformly to the simulation partitions. Higher frequency components closer to the viewer are allocated to partitions with finer grids and smaller time steps, while lower frequency components more far away from the viewer are allocated to partitions with coarser grids and larger time steps. As a result, the effective velocity field defined on each simulation partition is a mixture of frequency components, and the visible evolution of the mixed velocity can be sufficiently captured by the space-time resolution associated with the specific partition. To obtain the final solution, the effective velocity fields are solved semi-independently on different partitions, which provides richer visual details to the viewer in a more efficient way. Although the nested simulation partitions differ in size and resolution, they are all meshed with uniform rectangular grids, which makes the solver robust and efficient.

This novel view-dependent multiscale simulation framework distributes the computational resources according to the human visual perception of the target fluid motion, and is adaptive in both space and time

- *Y. Gao, B. Ren and S.M. Hu are with the Department of Computer Science, Tsinghua University, Beijing 100084, China.*
- *C.F. Li is with the College of Engineering, Swansea University, Swansea SA2 8PP, UK.*

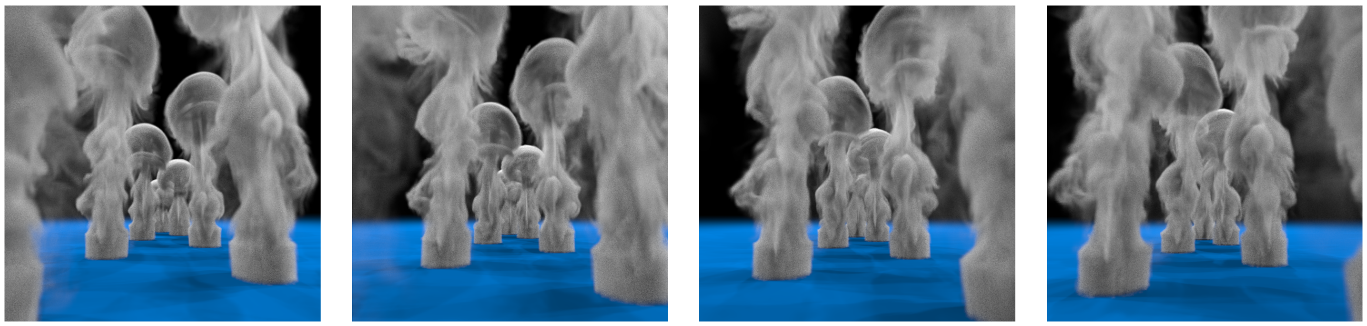

Fig. 1. Four snapshots of a view-dependent multiscale fluid simulation with moving camera positions. (6 partitions, grid sizes: $1/400$ – $1/100$, time steps: $1/120$ s – $1/30$ s)

dimensions. The main technical innovations include:

- Using a novel approach of space-filling curves, the EMD is efficiently extended to 3D and applied to decompose the velocity field into a small number of frequency components, which represent the fluid motion at different length scales.
- A spectrum based simulation pipeline is proposed, in which different frequency components evolve at different space-time resolutions. By doing so, it significantly reduces the numerical diffusion that causes damping of high frequency turbulence in previous methods, and preserves more fine-scale turbulence details in the result.
- The fluid domain is adaptively partitioned according to the camera position, and the fluid is simulated at different space-time resolution depending on its distance to the viewer. This approach considers both rendering and simulation together, and efficiently utilizes the computational resources in the places that most affect the final rendered result.

2 Previous Works

Jos Stam's unconditionally stable solver [2] made the grid-based fluid simulation popular in the graphics community. Since then, many different techniques have been developed to add details to the fluid. The basic approach is to reduce the numerical dissipation (also known as "numerical diffusion"). [3] used the vorticity confinement technique to prevent the rapid dissipation of vortices. [4] introduced artificial divergence sources to simulate gas explosion. [5] introduced vortex particles to add the vorticity more accurately. [6] introduced FLIP to overcome advection dissipation. Other methods including BFECC [7], QUICK [8], MacCormack [9] suggested using higher-order space discretization schemes and higher-order time integration schemes (e.g. Runge-Kutta methods). These methods discretize the whole fluid domain using uniform grids, thus they are all limited by the Nyquist frequency.

For 3D fluid simulation, a small increase of the grid resolution by a factor of k will cause a dramatic increase to the computational cost by a factor of k^4 [10]. Therefore, various techniques have been investigated in order to increase the simulation resolution while controlling the computational expense. [11], [12], [13], [14] proposed different methods to generate divergence free fields from random noise, and then used these artificial velocity fields to represent the turbulent flow. [15], [16], [17], [18], [19] simulated the fluid on a low-resolution grid to obtain the macro-scale flow, which was then combined with the artificial divergence-free velocity fields to mimic the turbulent flow at the micro scale. Instead of adding noise, [20] used the vortex particle method [5] to directly generate a high-resolution turbulent flow, and [13] synthesized the 3D velocity field from 2D slices. These synthesis methods do not perform high-resolution computation on the Navier-Stokes equations, and instead attempt to produce plausible results using artificial means. Thus, their results are nonphysical, but can be combined with any grid-based method.

Although grid-based fluid solvers are often preferred in the graphics community, other numerical schemes including finite volume [21], [22] and finite element methods [23] have also been used in many specific graphics applications. Some researchers have also exploited the viewing information in fluid simulations. Until recently, there have been mainly two types of approaches: (a) octree and adaptive mesh refinement methods [24], [25], which use non-uniform meshes to distinguish different levels of details for the fluids; and (b) multi-grid methods [8], [10], [26], which use multiple layers of meshes to represent fluid motions at different length scales. The idea of multi-grid simulations has also been adopted in the framework of smoothed-particle hydrodynamics to accelerate fluid simulation [27]. In a wider context, it is also noted that [28] presented a view-dependent multiscale simulation framework for fire simulations.

3 Algorithm Overview

Fluid phenomena are interesting and visually attractive because of turbulent fluid motions. It is well known in fluid dynamics that turbulence occurs and develops at different length and time scales, with the extent of scale difference indicated by the Reynolds number. In order to capture fine-scale features of turbulent flows, it is normally necessary to use fine simulation grids

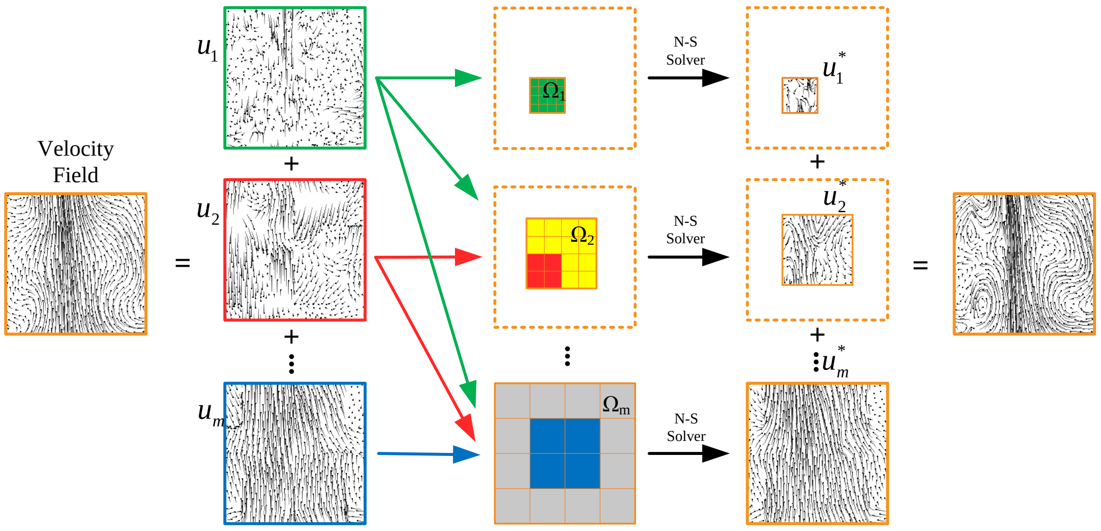

Fig. 2. The algorithm framework of view-dependent multiscale fluid simulation

and small time steps, or to use higher-order space discretization and time integration schemes. This creates a huge computational burden to the fluid simulator, in particular when simulating a large fluid domain. On the other hand, objects in a large scene are observed at different resolutions by human eyes depending on their distances to the viewer. Fine local fluid motions have a significant visual impact when the viewer is nearby, and as the distance to the viewer increases, these fine-scale features become less and less visible, while global fluid motions at larger scales becoming more and more dominant. Thus, for the purpose of achieving visually appealing results, there is a clear potential of benefit of utilizing the viewing information to improve the performance of fluid simulators.

We propose a view-dependent multiscale simulation framework as shown in Fig. 2. First, the fluid velocity field is decomposed into a series of frequency components $\mathbf{u}_1$, $\mathbf{u}_2$, $\cdots$, $\mathbf{u}_m$, representing the fluid motion at different length scales ranging from small to large. Next, according to the position of the viewer, m nested simulation partitions Ω_i are constructed, with Ω_1 indicating the vicinity of the viewer and Ω_m representing the whole fluid domain. These simulation partitions are all meshed into uniform rectangular grids, and each partition Ω_i is set with a different grid size depending on the length scale of the corresponding frequency component $\mathbf{u}_i$. Then, each frequency component $\mathbf{u}_i$ is sequentially allocated to partitions Ω_i, Ω_{i+1}, $\cdots$, Ω_m, such that for each partition, it only carries the component quantity that has not been supported by the previous ones. Thus, the effective velocity field $\mathbf{u}_i^*$ defined on partition Ω_i is a mixture of velocity components $\mathbf{u}_1$, $\cdots$, $\mathbf{u}_i$ that share a similar visual significance determined by their intrinsic length scales and distances to the viewer. To solve this combined velocity field $\mathbf{u}_i^*$ with uniform visual significance, a separate fluid simulation is performed on partition Ω_i with individually assigned grid size and time step. Finally, the total fluid motion in the whole fluid domain is constructed by adding up the results obtained on all simulation partitions. Depending on the fluid evolution, the velocity spectrum is repeatedly computed to ensure that the new fluid motion is efficiently represented by the frequency components $\mathbf{u}_1$, $\cdots$, $\mathbf{u}_m$.

The proposed view-dependent multiscale fluid simulation framework can be viewed as a multi-grid method combined with spectral decomposition. The idea of using spectral analysis in CFD applications is not entirely new, and a remarkable example is the large eddy simulation [29] that introduces spatial-temporal filters to reduce the range of length scales of the solution, hence reducing the computational cost. The feasibility of this new simulation framework relies on two assumptions: (a) the fluid velocity field can be decomposed into a small number of meaningful frequency components at different length scales; and (b) the Navier-Stokes equations can be linearized to allow separately solving each frequency component with varying grids and varying time steps. The consideration and solution of these two issues are addressed in Sections 4 and 5 respectively.

4 3D Velocity Field Decomposition

The fluid velocity field can be viewed as a time-varying signal defined in a 3D domain. From the viewpoint of physics, it is clear that the 3D velocity signal consists of intrinsic structures at different length scales. However, as turbulence is highly nonlinear and non-stationary, standard data analysis tools such as singular value decomposition [30], Fourier and wavelet analysis etc. typically produce many spurious frequency components causing energy spreading, which makes the resulting spectrum have little physical meaning. An exception is the empirical mode decomposition (EMD) [31], also known as Hilber-Huang transform, which was originally developed for processing nonlinear and non-stationary time series. Over the past decade, the EMD method has been extremely successful in engineering and successfully applied in various complicated data sets, including sea waves and earthquake signals etc.

For the sake of completeness, the standard EMD procedure is briefly reviewed in Section 4.1, after which it is extended into 3D cases in Section 4.2 for processing the fluid velocity field.

4.1 EMD Basics

The standard EMD method is designed for the analysis of one-dimensional signals, in particular time series. The main idea of EMD is to decompose the signal into a small number of intrinsic mode functions (IMF), which are based on and derived from the data. An IMF is any function with the same number of extrema and zero crossings, and with zero mean of the upper and lower envelops defined respectively by the local maxima and minima. From this definition, the IMF is a general oscillatory function, with possibly varying amplitude and frequency along the time axis. Thus, for representing signals, an IMF is much more powerful than the simple harmonic function, which has constant amplitude and frequency. Given a one-dimensional signal f, the EMD algorithm sequentially extracts its IMFs via a "sifting" procedure as follows:

1) Initialization $r_0 = f$, set index $k = 1$
2) Compute the k-th IMF, c_k
    a) Initialization $h_0 = r_{k-1}$, set index $j = 1$
    b) Find all local maxima and local minima of h_{j-1}
    c) Build the upper envelope $E_{max,j-1}$ by connecting all local maxima with a cubic spline, and build the lower envelope $E_{min,j-1}$ by connecting all local minima with a cubic spline
    d) Compute the mean of the upper and lower envelopes, $E_{mean,j-1} = \frac{1}{2}(E_{min,j-1} + E_{max,j-1})$
    e) $h_j = h_{j-1} - E_{mean,j-1}$
    f) If the IMF stopping criterion is satisfied, then $c_k = h_j$, else $j = j + 1$ and go to step 2(b)
3) $r_k = r_{k-1} - c_k$
4) If r_k is monotonic, the decomposition stops, else $k = k + 1$ and go to step 2

The signal f is decomposed as

$$f = \sum_{k=1}^{K} c_k + r_K, \tag{1}$$

where $c_i, i = 1, 2, .., K$ are the IMFs with the frequency ranging from high to low, and r_K is the residual.

For the IMF stopping criterion in step 2(f), different criteria have been suggested in the literature based on the definition of IMFs. In our applications, it is found that there is no visible difference in the final result if we simply fix the iteration number as 8 to 10. There is no rigorous convergence proof for the above algorithm, but practically it always converges very quickly [31]. The physical justification of the above EMD procedure is very solid and has been verified and validated in numerous experiments by various real data sets (see e.g. [32]).

4.2 3D EMD of Velocity Fields

The main challenge of extending the EMD into higher dimensional signals arises in the construction of the upper and lower envelopes (step 2(c) in the EMD algorithm). Unlike the simple closed-form solution of the 1D cubic spline interpolation, higher dimensional surface interpolation is complex and often involves time-consuming computation. For the 2D case, [33] and [34] introduced 2D radial basis functions and transformed the interpolation problem into a global optimization problem. It requires to solve a $m \times m$ linear system, where m is the total number of extrema. The associated computation is affordable for 2D image applications with hundreds of pixels along each axis, but is too slow for our 3D fluid simulations that require the EMD to be repeatedly performed in a large 3D space as turbulence develops. [35] proposed a fast bidimentional EMD algorithm, which is based on the Delaunay triangulation and cubic interpolation on triangles. In order to ensure the Delaunay triangulation to cover the whole domain, this method has to introduce a bunch of artificial extrema, and as a result it is not suitable for our 3D fluid simulations that require the highest level of automation and robustness. [36] tested a tensor-product based 2D EMD approach that applies separately 1D EMD on each row and column of an image, after which averaging the envelopes from different directions. Although it is much faster to do so, our experiments led to a similar conclusion as [36]: the result is generally worse in that each slice of data only contain a small portion of samples and the connection information contained in the original data has been seriously lost. As the EMD will be repeatedly performed in our fluid simulation framework, a more efficient and more robust 3D algorithm is needed.

We propose to use space-filling curves to flatten 3D data into 1D. First, a space filling curve is constructed to fill the fluid domain, and moving along the curve an index is assigned to each grid cell and saved in a template. Then, the 3D velocity field is rearranged into a 1D signal array according to the index template. Finally, the reshaped 1D signal is decomposed by using the 1D EMD algorithm, and the decomposition result is mapped back to the 3D space by using the same index template. In this simple 3D EMD approach, the EMD operation is essentially performed on the flattened 1D data set, and therefore it converges in the same way as the standard 1D EMD method [31]. As the index template of space-filling curve can be pre-computed, the CPU expense of the 3D EMD is essentially the same as 1D EMD, which is linearly proportional to the sampling density. Owing to the analytic cubic spline interpolation, our space-filling curve EMD technique is extremely fast. For 2D cases, we have compared with the RBF method. It is

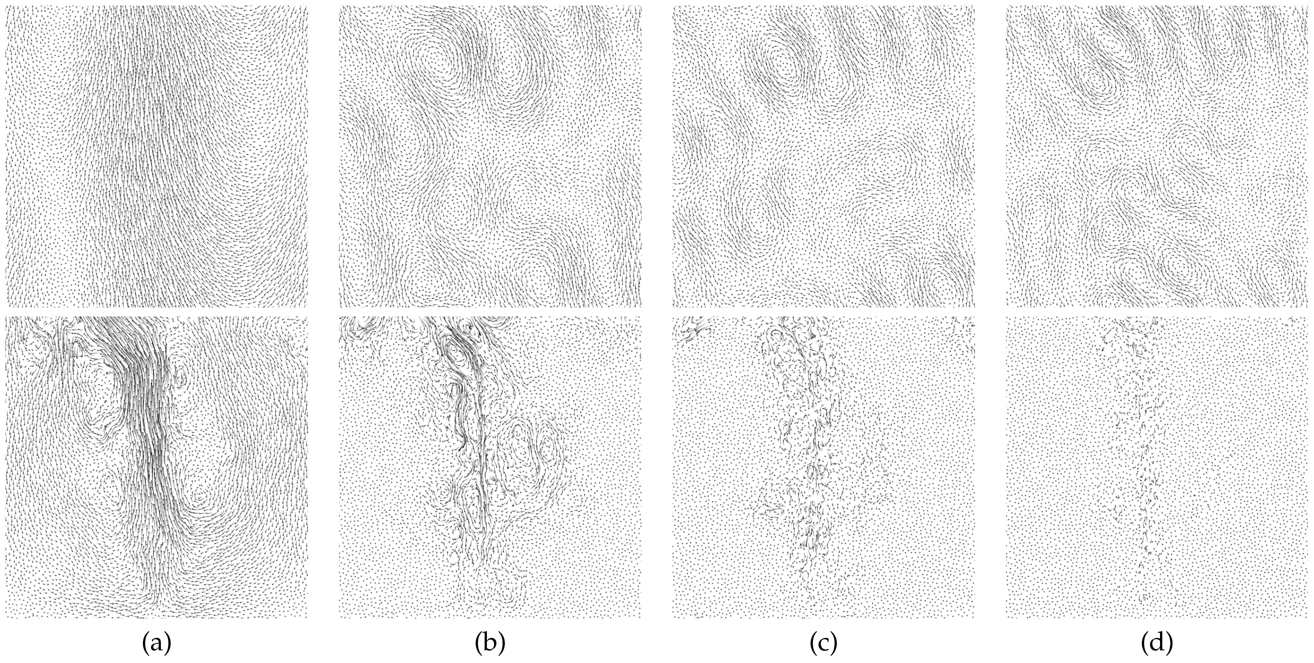

Fig. 3. Decomposition comparison of a 2D fluid velocity field. The top row shows Fourier decomposition, and the bottom row shows the EMD result. (a)-(d) are the 1st, 3rd, 4th, and 5th components from low frequency to high frequency. The EMD and Fourier results are obtained with the same sampling resolution.

found that our approach is at least 20 times faster in all test examples, and the new method also provides better accuracy because it avoids the numerical error caused by the least squares approximation required in the RBF approach. In the context of fluid simulation, the CPU cost of an individual 3D EMD is about half of a single pressure solver executed on the same sampling grid.

Different space-filling curves have been tested, including the Hilbert curve, the Z-order curve, the Koch curve and the Gosper curve. In 2D cases, the Hilbert curve and the Z-order curve are found to have boundary artifacts caused by their regular quad fractal structures. By using Koch or Gosper curves, the boundary artifacts can be effectively removed. In 3D cases, all four curves give good decomposition results without visible discontinuities. The reason is that both the 3D velocity filed and the 3D space filling curves are sufficiently complex to avoid the development of boundary artifacts. For the sake of simplicity, we use the Koch curve for 2D examples and the Hilbert curve for 3D examples in this paper. It is noted that the Z-order curve has recently been in SPH simulations to compute SPH neighborhoods rapidly [37], which also demonstrates the benefit of using space-filling curves to accelerate 3D data processing.

The 3D Hilbert curve is defined on a cube, and when using the n-th approximation to the limiting curve, the length of the curve is 2^n. However, the fluid domain is not necessarily a cube. Therefore, we build the Hilbert curve with the smallest n such that $2^{n/3}$ is greater than the maximum velocity resolution in x-, y- and z-directions. When moving along the Hilbert curve, the cell index is increased and saved if and only if the current

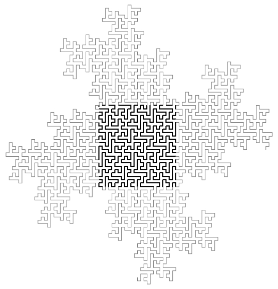

Fig. 4. Quadric Koch Curve.

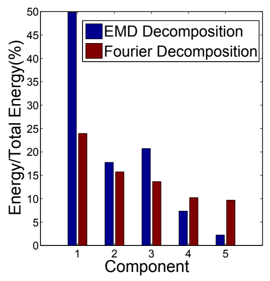

Fig. 5. Energy distribution of the lowest 5 frequency components.

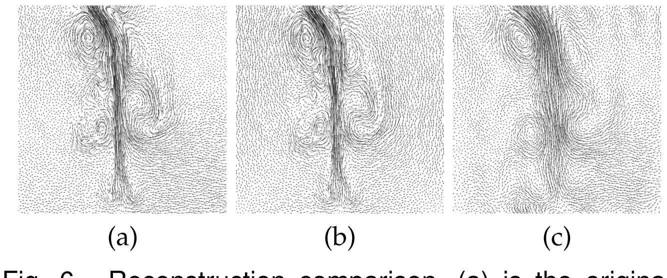

Fig. 6. Reconstruction comparison. (a) is the original velocity field, (b) is the sum of the first 5 EMD components and (c) is the sum of the first 5 Fourier components.

position is located in the fluid domain. A similar method is applied to the Koch curve in 2D cases. In Fig. 4, the gray line is the whole Koch curve and the bold black line is the space filling curve we used. This strategy preserves as much as possible the locality of the space filling curve.

To flatten 3D data into 1D for EMD operations is an approximate treatment, and so doing inevitably causes some loss of local connectivity information presented in the original 3D data. For use in fluid simulation, we have tested the new space-fill curve EMD approach in numerous examples, both in 2D and 3D. Fig. 3 is an 2D example of our EMD result compared with Fourier decomposition. The velocity field is generated by using a 2D grid solver and for fair comparison, the same sampling resolution is used in the EMD and Fourier decomposition. The comparison shows that the EMD method retains better locality and has better efficiency in terms of the number of terms required to represent the original velocity field. The EMD frequency components concentrate in the areas where turbulence occurs, while the Fourier components have ring-shape vortices everywhere in the fluid domain, which is non-physical. Fig. 5 shows the energy distribution of the frequency components obtained in the EMD method and the Fourier decomposition. It is clear that fewer EMD components are needed in order to recover the same amount of energy for the fluid motion. A direct reconstruction comparison is given in Fig. 6 (please zoom in to see the difference), where Figs. 6(a-c) show respectively the original fluid velocity field, the EMD and the Fourier reconstructions using the same number of components. It can be seen that by using just five IMFs, the EMD method perfectly recovered the original velocity field with no visible defects, while a large amount fine-scale details are lost in the Fourier reconstruction.

For decomposing the fluid velocity field, an added benefit of the EMD method is on dealing with objects presented inside the fluid domain, where the fluid velocity field can be discontinuous on the object boundary. Most standard data analysis tools use functional basis with fixed amplitudes and frequencies, and consequently the signal discontinuity will cause many spurious frequency components due to energy spreading. However, the functional basis of the EMD method is adaptively determined by the local features of the signal. The IMFs have varying amplitudes and frequencies, so that the energy spreading caused by signal discontinuity is minimized. Indeed, this is one of the major advantages of the EMD technique [31].

Using the EMD method, the velocity field $\mathbf{u}$ in the simulation domain is represented as:

$$\mathbf{u} = \sum_{i=1}^{m} \mathbf{u}_i \tag{2}$$

where $\mathbf{u}_i$, $i = 1, 2, \cdots, m$, are frequency components representing fluid motions at different length scales, ranging from small to large. In our implementation, m is a user specified constant controlling how many IMFs to be extracted from the velocity field. Thus, $\mathbf{u}_i$, $i = 1, \cdots, m-1$, are IMF components, and $\mathbf{u}_m$ is a non-IMF component. As $\mathbf{u}_m$ consists of all lower frequency tail IMFs and the residual term, it carries the majority of the kinetic energy of the fluid flow. Benefited from the adaptive and data dependent nature of IMFs, the nonlinear and non-stationary fluid velocity field can be effectively represented with a small number of frequency components. In our limited experiments, 5 to 8 frequency components are sufficient to represent the velocity fields. Note that the EMD is performed separately for x-, y- and z- directions, and then adding them together to obtain the vector-valued decomposition (2).

5 View Dependent Multiscale Simulation

For incompressible ideal fluids, the Navier-Stokes equations are:

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mathbf{f}, \tag{3}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4}$$

where $\mathbf{u}$ is the velocity, p the pressure, ρ the fluid density, and $\mathbf{f}$ the effective body force including gravity, buoyancy and vorticity confinement etc.

The human visual perception of a large dynamic fluid scene has two main features:

- Fluid motions are observed at different resolutions by human eyes, depending on the distance from the viewer to the location where the motion is developing. The smaller the distance is, the higher resolution will be received; and vice versa.
- The fluid motion consists of intrinsic structures, i.e. frequency components, evolving at different length and time scales. These multiscale frequency components generate unequal visual impacts. When the viewer is nearby, the fast-developing small scale components are more significant in our observation; and when the viewer is at distance, the slow-moving large scale components become more dominant.

In order to achieve the best visual effects with the minimum computational cost, the fluid solver needs to take into account *both* of the above aspects. This is done by integrating spectral analysis and domain partition into a view driven simulation framework, whose details are explained in the following subsections.

5.1 Dynamics of Multiscale Flow

In the space dimension, the multiscale motion components of a turbulent flow are revealed in Eqn. (2) by using the EMD method. Substituting Eqn. (2) into the N-S equations (3 - 4) and setting the fluid density to unit yields:

$$\sum_{i=1}^{m} \frac{\partial \mathbf{u}_i}{\partial t} + \sum_{i=1}^{m} (\mathbf{u} \cdot \nabla)\mathbf{u}_i = -\nabla p + \mathbf{f}, \tag{5}$$

$$\sum_{i=1}^{m} \nabla \cdot \mathbf{u}_i = 0. \tag{6}$$

When an explicit solver is adopted, the total fluid velocity $\mathbf{u}$ is computed using the results from the previous time steps, thus $\mathbf{u}$ can be considered as semi-decoupled from $\mathbf{u}_i$ in Eqn. (5).

Eqns. (5 - 6) show that multiscale fluid motions are coupled together to satisfy momentum and mass conservation. However, from the viewpoint of physics [38], fluid motions $\mathbf{u}_i$ differ not only in their length scales, but they also develop at different pace in the time dimension, with micro-scale motions developing fast and macro-scale motions developing relatively slow. Thus, if the observation is fixed to a small window T in the time axis, the inter-frequency exchange of momentum and mass can be ignored, and this leads to

$$\frac{\partial \mathbf{u}_i}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}_i = -\nabla p_i \quad i = 1, 2, \cdots, m-1, \tag{7}$$

$$\frac{\partial \mathbf{u}_m}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}_m = -\nabla p_m + \mathbf{f}, \tag{8}$$

$$\nabla \cdot \mathbf{u}_i = 0 \quad i = 1, 2, \cdots, m, \tag{9}$$

where p_i, $i = 1, 2, \cdots, m$ are the unknown fluid pressure corresponding to the motion components $\mathbf{u}_i$. Typical body forces, such as gravity and buoyancy, change much slower comparing to the rapid development of micro-scale turbulent motions. This is particularly true for ideal fluids [38], whose viscous force is zero and Reynolds number is infinity. Therefore, in the momentum Eqn. (7), the influence from the slow changing body forces to the fast developing micro-scale fluid motions is also ignored, and the body force is only included in Eqn. (8) for the mixed low-frequency component $\mathbf{u}_m$. By allowing all body forces to directly work on the $\mathbf{u}_m$ motion, the dominant energy carrier obtained in the EMD (2), the energy transfer process occurring at the macro-scale level is emphasized. However, if fast changing body forces are involved, they should be likewise decomposed and applied to the corresponding velocity component.

5.2 View-Dependent Simulation of Multiscale Flow

Eqns. (7 - 9) describe the dynamics of multiscale flow. Our aim is to solve these equations according to the camera settings such that all visible fluid motions at both micro- and macro- scales are accurately captured with the minimum computational cost.

First, the fluid domain is divided into m nested partitions Ω_i, $i = 1, 2, \cdots m$ such that $\Omega_1 \subset \Omega_2 \subset \cdots \subset \Omega_m$, where Ω_m represents the whole fluid domain. These nested partitions are all centered with respected to the view frustum, so that partitions Ω_i, $i = 1, 2, \cdots, m$ provide a natural indication for the distance between the viewer and the fluid point, ranging from small to large. It is noted that by building the partitions Ω_i with respect to the view frustum, the view direction and view angle are also taken into account. As the viewer-to-fluid distance increases, the visibility of the fluid motion drops, which sequentially reduces the accuracy requirement of the simulation. Therefore, these simulation partitions are discretized using different grid sizes and time steps, and with the increase of index i, the space-time resolution of Ω_i decreases. In particular, the grid size and time step of each partition Ω_i are set to allow an economical and yet sufficiently accurate description of the motion $\mathbf{u}_i$.

Next, depending on the viewer-to-fluid distance, each motion component $\mathbf{u}_i$ is adaptively represented at different space-time resolutions. This is achieved by distributing the velocity quantities of $\mathbf{u}_i$ to partitions Ω_j, $j = i, i+1, \cdots, m$ such that the motion $\mathbf{u}_i$ is discretized on a composite grid $\Omega_i \cup \{\Omega_{i+1} - \Omega_i\} \cup \cdots \cup \{\Omega_m - \Omega_{m-1}\}$. As shown in Fig. 2, after all frequency components $\mathbf{u}_i$ have been distributed to the simulation partitions, the velocity field on each partition Ω_i becomes a composite field $\mathbf{u}_i^*$ as follows:

$$\mathbf{u}_i^* = \begin{cases} \mathbf{u}_i & for \ \ \Omega_{i-1} \\ \sum_{j=1}^{i} \mathbf{u}_j & for \ \ \Omega_i - \Omega_{i-1} \end{cases} \tag{10}$$

The effective velocity $\mathbf{u}_i^*$ collects all visible fluid motions measured at the space-time resolution of Ω_i.

Then, reorganizing Eqns. (7 - 9) according to Eqn. (10) yields:

$$\frac{\partial \mathbf{u}_i^*}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}_i^* = -\nabla p_i^* \quad i = 1, 2, \cdots, m-1, \tag{11}$$

$$\frac{\partial \mathbf{u}_m^*}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}_m^* = -\nabla p_m^* + \mathbf{f}, \tag{12}$$

$$\nabla \cdot \mathbf{u}_i^* = 0 \quad i = 1, 2, \cdots, m, \tag{13}$$

where p_i^*, $i = 1, 2, \cdots, m$ are the unknown fluid pressure corresponding to the composite velocity components $\mathbf{u}_i^*$. Although similar in formulation, it should be noted that Eqns. (11 - 13) and Eqns. (7 - 9) describe *totally different* physical phenomena. Eqns. (11 - 13) are defined on partitions Ω_i, $i = 1, 2, \cdots, m$ respectively, and for each partition Ω_i, they describe the evolution of all fluid motions that are visible at the space-time resolution associated with Ω_i. Eqns. (7 - 9) are defined in the whole fluid domain, and they describe the dynamics of the fluid motion at each individual length scale, regardless of its visibility to the viewer.

Finally, the fluid simulation is performed by solving the Eqns. (11 - 13) on nested partitions Ω_i, $i = 1, 2, \cdots, m$ respectively. The initial values of $\mathbf{u}_i^*$ are computed with Eqn. (10), in which the frequency components $\mathbf{u}_i$ are obtained from the EMD (2). Starting from $i = 1$ and going through each simulation partition Ω_i, the solution $\mathbf{u}_i^*$ is obtained by using the standard advection-projection scheme [2]. Specifically, the advection step solves equation

$$\frac{\partial \mathbf{u}_i^*}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}_i^* = 0. \tag{14}$$

Note that the background velocity field for advection is the total velocity $\mathbf{u}$ instead of the velocity component $\mathbf{u}_i^*$. Similarly, the projection step solves equations

$$\frac{\partial \mathbf{u}_i^*}{\partial t} = -\nabla p_i^*, \tag{15}$$

$$\nabla \cdot \mathbf{u}_i^* = 0. \tag{16}$$

Note that for the last partition Ω_m, the external body force $\mathbf{f}$ is added into Eqn. (15). For the pressure solver, we use the standard preconditioned conjugate gradient method with the preconditioner obtained through the incomplete Cholesky decomposition. The final solution of the fluid is:

$$\mathbf{u} = \mathbf{u}_1^* \oplus \mathbf{u}_2^* \oplus \cdots \oplus \mathbf{u}_m^* \tag{17}$$

where $\oplus$ denotes the superposition of velocity fields $\mathbf{u}_i^*$ defined in different partitions. As Eqns. (11 - 13) hold only when the observation is fixed in a relatively small time window T, the EMD operation (2) needs to be repeatedly performed after certain time steps to re-initialize the solution process (14 - 17). This EMD re-initialization step is necessary to ensure an adequate and timely capture of the cross-scale motion transfer of the fluid.

Boundary conditions: For internal partitions Ω_i, $i = 1, \cdots, m-1$, the boundary conditions are set as $\mathbf{u}_i^* = 0$ on $\partial\Omega_i$. For the partition Ω_m, the real boundary condition of the whole fluid domain is used on $\partial\Omega_m$. These simplifications practically over restrict the energy exchange between partitions. By doing so, we sacrifice the accuracy in order to minimize the coupling between partitions and improve the efficiency of obtaining visually plausible results. Our method also supports internal boundaries. For static obstacles in the fluid domain, each velocity component deals with the obstacle in the same way as the traditional methods, e.g. using simple obstacle discretization or some more precise models. As the obstacle is static, the final velocity field automatically satisfies the non-slip boundary condition. For dynamic obstacles, a practical approach is to add the dynamic boundary condition to the lowest frequency component, while adding static boundary conditions to all the other components.

5.3 Computational Issues

Our multiscale fluid simulation is driven by the viewer. In standard rendering systems, such as the PBRT [39] used in this work, the fluid domain is defined in the object space, then transformed into the view space by model and view matrices, and finally projected into the image space according to camera parameters (projection matrix and viewport). We integrate the inverse of this pipeline into our simulator to control the levels of details in the simulation.

The fluid domain is divided into simulation partitions according to the distance to the viewer and the view direction, and each partition is individually assigned with a grid size and a time step. Thus, the partitions move when the viewer moves, which then requires the fluid velocity to be transferred between grids of different sizes. For simplicity, we use linear interpolation for the velocity transfer between coarse grids and fine grids.

In the current implementation, the grid sizes and time steps are manually set by the user based on the size of simulation domain, the camera setting and the characteristics of IMFs. Separate velocity components communicate with each other through the advection term (14) and the EMD re-initialization. It is possible to automatically determine the spatial-temporal resolution. Specifically, the grid size can be associated to the dimension of the simulation domain and the spatial frequencies of IMFs, which can be obtained via Hilbert transforms. Once the grid size is fixed, the corresponding time step can then be determined in conjunction with the camera motion. This important adaptivity aspect will be pursued in our future work as detailed in Section 7.

A direct application of frequency decomposition is modulating the velocity filed. In many cases, the animator wants to add turbulence into the fluid. One way of doing so is to boost the high frequency components when calculating the final velocity. However, this simple approach makes the solver unstable because a positive feedback loop could be formed and causes the solver to crash. Hence, we first calculate the average energy of each frequency component, and decrease the velocity of the low frequency components according to the energy increment of the high frequency components. Under this energy conservation constraint, the modulating process becomes robust. Another safe modulating approach is to change the vorticity confinement coefficients for each components. As demonstrated in [3], setting the vorticity confinement larger will not only enhances the vortices but also affects the behavior of the whole fluid. We found that by boosting the vorticity confinements only in the high frequency components, the result shows more vortices in the fluid as well as maintains the basic fluid motion.

Given a target fluid and the camera settings, the view-dependent multiscale fluid simulation is performed as follows:

1) Generate a Hilbert curve to cover the whole fluid domain and build the 3D-to-1D index template
2) Compose an ordered work list consisting of four types of jobs: partition, EMD, simulation and output
3) Follow the work list to do
    - For partition request: according to the current camera settings, the whole fluid domain is divided into simulation partitions Ω_i with fixed grid sizes and time steps
    - For EMD request: compute the velocity spectrum (2) with 3D EMD
    - For simulation request: solve Eqns. (14 - 16) on the specific simulation partition Ω_i
    - For output request: output the current velocity field $\mathbf{u}$

In step (2), time entries of the partition request are determined according to the camera motion; time entries of the EMD request are set with a fixed time interval specified by the user; time entries of the simulation request are calculated according to the fixed time step of

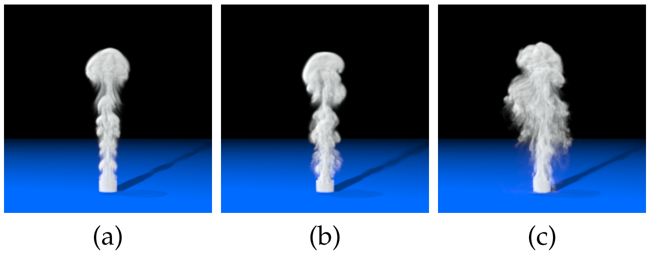
Fig. 7. Comparison of the standard solver and our new method without view-control. (a) standard method, (b) our new method, and (c) out method with editing

each simulation partition; and time entries of the output request are set according to the animation requirement. In the current implementation, the oldest time step is always executed first in order to get the most up-to-date information from the other fluid simulations. Also, in the advection step, we simply use the latest total velocity field as the background velocity. By doing so, we ignore the numerical error caused by the simulations being out of synchronization. The main computational cost of our simulation framework is in the advection-projection solutions, which are performed separately on different partitions with different space-time resolutions. As high resolution solutions are only performed for the closest partitions to the viewer, usually very small domains, our simulation runs much faster comparing to the standard N-S solver using a uniform high resolution grid.

Comparing with octree and adaptive mesh refinement methods, the proposed method differ mainly in two aspects: 1) We distinguish the fluid flow not only by its distance to the viewer (resolved by setting multiple simulation partitions), but also by its intrinsic motions at different length scales (resolved by EMD). Both spatial and temporal resolutions are adaptive in our method, while the octree and AMR approaches are often adaptive only in the space dimension. 2) Octree and ARM methods use non-uniform grids, and we use multiple partitions meshed into uniform grids. The use of uniform grids and simple data structures significantly simplifies the implementation and computational complexity. In a wider sense, the new method can be viewed as a multi-grid approach combined with spectral analysis. Unlike other multi-grid methods using prefixed simulation resolutions independent to the evolution of fluid flows, the space-time resolutions for different simulation partitions are determined according to the spectral decomposition result of the fluid velocity field. Therefore, the new method is more adaptive, and can support moving camera positions and developing fluid flows in a uniform framework.

6 Result

Several experiments are presented in this section to demonstrate the performance of the new fluid simulation framework (see Table 1). All numerical simulations are

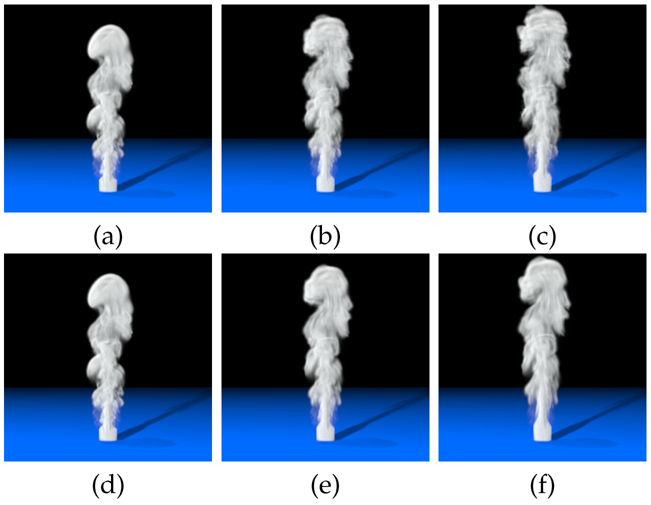
Fig. 8. Comparison of a low-frequency velocity field solved on fine and coarse grids. The same low-frequency flow is solved respectively on a fine grid and a coarse grid, where (a), (b) and (c) are the fine-grid results from the 1st, 8th and 16th frames, and (d), (e) and (f) are the corresponding coarse-grid results.

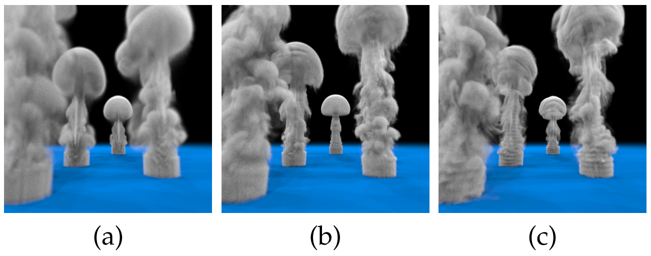
Fig. 9. Comparison of the standard method and our new method with view-control. (a) standard method, and (b) and (c) our new method using different EMD intervals.

performed on a PC platform with an Intel Core2 2.4 GHz CPU and 8 GB memory.

The first example compares the new method without view-control and the standard grid-based N-S solver. Fig. 7(a) is the result obtained from the standard solver on a $128 \times 256 \times 128$ grid. Fig. 7(b) is the result obtained

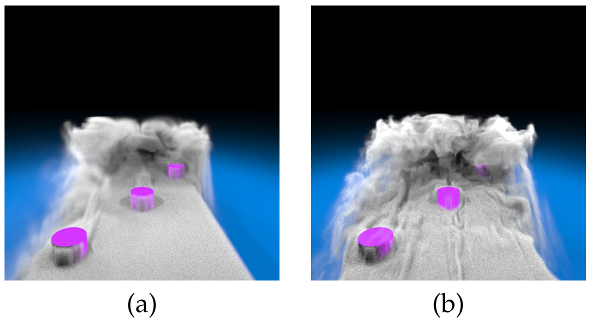
Fig. 10. Comparison of the standard solver and our new method with a static obstacle in the domain. (a) standard method, (b) our new method using 4 partitions without view control.

Parameter	Fig. 7(a)	Fig. 7(b)	Fig. 7(c)	Fig. 9(a)	Fig. 9(b)	Fig. 9(c)	Fig. 10(a)	Fig. 10(b)	Fig. 1
highest resolution	1/128	1/128	1/128	1/200	1/400	1/400	1/64	1/128	1/400
avg. time per frame	45.1s	33.6s	35.5s	243.6s	228.9s	266.3s	56.7s	69.2s	328.0s
speedup percentage	N/A	25.5%	23.3%	N/A	6%	-9.3%	N/A	-22%	N/A
component num.	N/A	6	6	N/A	5	5	N/A	5	5
max down sample rate	N/A	2	2	N/A	4	4	N/A	4	4
view control	N/A	No	No	N/A	Yes	Yes	N/A	Yes	Yes
EMD interval	N/A	10	10	N/A	10	15	N/A	20	5
editing	N/A	No	Yes	N/A	No	No	N/A	No	No

TABLE 1
Simulation parameters and performance.

using the new method with six simulation partitions all covering the whole fluid domain. The first five partitions are meshed as $128 \times 256 \times 128$, and the last partition is meshed as $64 \times 128 \times 64$. All six partitions use the same time step in their simulations, and the EMD is performed every 10 frames to update the velocity spectrum. Comparing Figs. 7(a-b), it is observed that the new method produces the correct result with more fine-scale features. This is because the new method separates different frequency components and solves them on different partitions, and by doing so, the numerical dissipation is effectively reduced. For each time step, the average CPU time cost is 45.1 seconds in the standard N-S solver, and 33.6 for our new method. Without view control, the new method is about 25% faster than the standard grid solver. This is because (a) the sixth partition is solved on the coarse grid; and (b) the first five partitions do not have any body force and as a result, their simulations converge very fast, typically within 5 iterations. However, the standard N-S solver uses the fine grid and in each time step it takes 50 - 80 iterations to converge to the error threshold 10^{-5}. Fig. 7(c) shows a frequency editing result, where the first three components are amplified by a factor 5, the next two components by 2, and the last component accordingly attenuated. It can be seen that more fine-scale turbulence details are achieved while the global motion of the plume (dominated by the lower frequency components) is accurately retained, which is important for practical editing.

Our new method assumes that the low frequency components of a velocity field can be simulated on coarse grids to save computational cost. This is verified in Fig. 8, where a plume is developed in a velocity field that only contains low frequency components. Figs. 8(a-c) are the results obtained using a fine grid ($128 \times 256 \times 128$), and Figs. 8(d-f) are solved on a coarse grid ($64 \times 128 \times 64$). The two groups of results are very similar up to frame 8 and they become more different as the simulation continues. This observation confirms that the low frequency components of a velocity field do generate high frequency motions as time goes, but these newly generated high frequency components are neglectable in the beginning period, during which the fluid motion can be well captured by using a coarse grid. Therefore, we choose different grid resolutions to economically simulate different frequency components, and periodically recompute the velocity spectrum to adjust the frequency-component allocation and ensure that every frequency component is always simulated using the right grid resolution.

The third example examines the effect of view-dependent partitioning. The viewpoint is fixed inside the fluid domain. Fig. 9(a) shows the result obtained using the standard N-S solver on a $200\times200\times400$ grid (grid size $1/200$). It can be seen that all four plumes are captured at the same resolution, and the nearest plume to the viewer is lack of fine-scale details, which makes the scene look unnatural. Fig. 9(b) shows the result obtained using the new method. It can be seen that different levels of details are obtained in the scene, with more fine-scale features captured for plumes closer to the viewer. Five simulation partitions with different dimensions and different grid sizes are used in the simulation. The first four partitions only cover approximately half depth of the scene and the fifth partition covers the whole fluid domain. The grid sizes are $1/400$, $1/400$, $1/200$, $1/200$ and $1/100$, respectively. As small grid sizes are only used on small partitions while larger partitions using larger grid sizes, the total memory used in the new method is similar to the standard solver. Fig. 9(c) shows the result of the new method using another set of parameters. In contrast to (b), the forth partition covers the whole domain, and the EMD interval increases to 15. Fig. 9(c) has some stair-like artifacts at the bottom of each plume. These artifacts are partially caused by the larger interval between adjacent EMD operations. Specifically, at a fixed frequency, the smoke source is added into the simulation as a cylindrical smoke density field together with some buoyancy forces. As the buoyancy force is only added to the lowest frequency component in our simulation framework (see Eqn.(8)), the smoke source is directly linked to the lowest frequency component. When the EMD interval increases, it takes longer to mix the velocity fields at different frequencies. Thus, the newly added smoke mainly moves with the lowest frequency component at the initial stage, and produces some stair-like artifacts. This kind of artifacts can also be observed in the standard solver when a strong smoke source is used in the simulation. In general, this type of artifacts can be effectively removed by adjusting the strength of the smoke source. For each time step, the average CPU time cost for the standard solver is

243.6 seconds, and 228.9 and 266.3 seconds for the new method. As shown in Table 1, by using different time steps in different partitions, the new method doubled the simulation resolution with essentially no speed hit.

The fourth example shows a static obstacle in the simulation domain. Fig. 10(a) is the result of the standard solver on a $64 \times 64 \times 256$ grid (grid size $1/64$). The relatively low resolution makes the result look flat. Fig. 10(b) is the result of our method using five partitions of which the last two partitions cover the whole fluid domain. The grid sizes from high frequency to low frequency are $1/128$, $1/128$, $1/128$, $1/64$, $1/32$. In Fig. 10(b), the non-potential flow structures observed at the upwind of the cylinders are caused by the interaction of the closely placed cylinders and the non-slip boundary effect. These high frequency structures are not resolved by the standard solver (Fig. 10(a)) due to numerical dissipation on the coarser mesh (grid size 1/64), while they are emphasized in the proposed solver (Fig. 10(b)) because the high frequency motion is solved separately on a finer mesh (grid size 1/128). The average CPU time costs of the standard method and our new method are 56.7 seconds and 69.2 seconds respectively. Again, it can be seen that with a similar computational cost, the new method doubled the simulation resolution producing more fine-scale details consistent with real-life observations.

The last example demonstrates the new method in a simulation with a moving viewpoint. Fig. 1 shows the simulation result, which is computed on six moving partitions. The maximum grid size is $1/100$, and the minimum is $1/400$. The maximum time step is $1/30$ s, which is used on the largest partition, and the minimum time step is $1/120$ s, which is used on the smallest partition. The space-time resolution of each partition is fixed, but its position and dimension change automatically as the viewpoint moves. It can be seen that the new method is robust and efficient, and it provides natural-looking results with multiple levels of details that are consistent with human visual perception.

7 Conclusion and Limitation

We propose a view-dependent multiscale fluid simulation framework that exploits both the viewing information in human visual perception and the multiscale velocity spectrum of a turbulent flow. In the new simulation framework, the fluid is solved at different space-time resolutions according to its visual impact. Specifically, high-resolution simulations are performed for the fluid regions closer to the viewer and for the frequency components more visible to human eyes, and vice versa. The new simulator better utilizes the computing power such that (a) for the same simulation task, it is faster than the traditional grid-based N-S solver; and (b) with the same computational resources (CPU time and memory storage), it can simulate a larger fluid scene or produce richer fine-scale details. Also, as the multiscale fluid motions are distinguished in our simulation, the numerical dissipation is effectively reduced. In particular, by modulating the simulation in the frequency space of the fluid motion, the new simulator provides the animator a simple way to edit and enhance the visual effects of fluids.

The current implementation does not allow moving internal obstacles. However, the extension to cope with moving internal objects is relatively straight forward, and care must be taken when the object moves across the boundary of simulation partitions. The main limitation of the proposed new framework is in three folds:

- For a fluid scene observed at multiple viewpoints, the simulation partitions become irregularly shaped, depending on the relative positions of different viewpoints. The complicated geometry of simulation partitions make the grid-based solver more complex in implementation, and a finite-volume solver might then become a better option.
- The proposed view-dependent simulation framework requires the camera to move continuously without jump. Discontinuous camera positions will cause the algorithm lose its advantage of capturing fine-scale details. This is because fine-scale motions are only simulated in the neighborhood of the previous camera focus instead of the whole fluid domain. Similarly, the performance can potentially drop with rapidly moving cameras because a larger buffer area will be needed for high resolution partitions.
- Our current implementation is purely sequential. As the simulations running on different partitions are relatively independent, the algorithm can be readily parallelized. Furthermore, as each partition is meshed into uniform grids, its simulation can also be accelerated with GPU implementation.

These three important aspects will be pursued in our future work.

References

[1] A. Oliva, A. Torralba, and P. G. Schyns, "Hybrid images," *ACM Trans. Graph.*, vol. 25, pp. 527–532, July 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141919

[2] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128. [Online]. Available: http://dx.doi.org/10.1145/311535.311548

[3] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 15–22. [Online]. Available: http://doi.acm.org/10.1145/383259.383260

[4] B. E. Feldman, J. F. O'Brien, and O. Arikan, "Animating suspended particle explosions," *ACM Trans. Graph.*, vol. 22, pp. 708–715, July 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882336

[5] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *ACM Trans. Graph.*, vol. 24, pp. 910–914, July 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073282

[6] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Trans. Graph.*, vol. 24, pp. 965–972, July 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073298

[7] T. F. Dupont and Y. Liu, "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function," *Journal of Computational Physics*, vol. 190, pp. 311–324, 2003.

[8] J. Molemaker, J. M. Cohen, S. Patel, and J. Noh, "Low viscosity flow simulations for animation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 9–18. [Online]. Available: http://portal.acm.org/citation.cfm?id=1632592.1632595

[9] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, "An unconditionally stable maccormack method," *J. Sci. Comput.*, vol. 35, no. 2-3, pp. 350–371, 2008.

[10] M. Lentine, W. Zheng, and R. Fedkiw, "A novel algorithm for incompressible flow using only a coarse grid projection," *ACM Trans. Graph.*, vol. 29, pp. 114:1–114:9, July 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778851

[11] J. Stam and E. Fiume, "Turbulent wind fields for gaseous phenomena," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '93. New York, NY, USA: ACM, 1993, pp. 369–376. [Online]. Available: http://doi.acm.org/10.1145/166117.166163

[12] A. Lamorlette and N. Foster, "Structural modeling of flames for a production environment," *ACM Trans. Graph.*, vol. 21, pp. 729–735, July 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566644

[13] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw, "Smoke simulation for large scale phenomena," *ACM Trans. Graph.*, vol. 22, pp. 703–707, July 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882335

[14] R. Bridson, J. Houriham, and M. Nordenstam, "Curl-noise for procedural fluid flow," *ACM Trans. Graph.*, vol. 26, July 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276435

[15] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," *ACM Trans. Graph.*, vol. 27, pp. 50:1–50:6, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360649

[16] R. Narain, J. Sewall, M. Carlson, and M. C. Lin, "Fast animation of turbulence using energy transport and procedural synthesis," *ACM Trans. Graph.*, vol. 27, pp. 166:1–166:8, December 2008. [Online]. Available: http://doi.acm.org/10.1145/1409060.1409119

[17] H. Schechter and R. Bridson, "Evolving sub-grid turbulence for smoke animation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 1–7. [Online]. Available: http://portal.acm.org/citation.cfm?id=1632592.1632594

[18] T. Pfaff, N. Thuerey, A. Selle, and M. Gross, "Synthetic turbulence using artificial boundary layers," *ACM Trans. Graph.*, vol. 28, pp. 121:1–121:10, December 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618467

[19] T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, and M. Gross, "Scalable fluid simulation using anisotropic turbulence particles," *ACM Trans. Graph.*, vol. 29, pp. 174:1–174:8, December 2010. [Online]. Available: http://doi.acm.org/10.1145/1882261.1866196

[20] J.-C. Yoon, H. R. Kam, J.-M. Hong, S.-J. Kang, and C.-H. Kim, "Procedural synthesis using vortex particle method for fluid simulation," *Comput. Graph. Forum*, vol. 28, no. 7, pp. 1853–1859, 2009.

[21] P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun, "Energy-preserving integrators for fluid animation," *ACM Trans. Graph.*, vol. 28, pp. 38:1–38:8, July 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531344

[22] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Trans. Graph.*, vol. 26, January 2007. [Online]. Available: http://doi.acm.org/10.1145/1189762.1189766

[23] B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," *ACM Trans. Graph.*, vol. 24, pp. 904–909, July 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073281

[24] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Trans. Graph.*, vol. 23, pp. 457–462, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015745

[25] J. Kim, I. Ihm, and D. Cha, "View-dependent adaptive animation of liquids," *ETRI Journal*, vol. 28, pp. 697–708, December 2006.

[26] M. B. Nielsen, B. B. Christensen, N. B. Zafar, D. Roble, and K. Museth, "Guiding of smoke animations through variational coupling of simulations at different resolutions," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '09. New York, NY, USA: ACM, 2009, pp. 217–226. [Online]. Available: http://doi.acm.org/10.1145/1599470.1599499

[27] S. Barbara and M. Gross, "Two-scale particle simulation," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 72:1–72:8, 2011.

[28] C. Horvath and W. Geiger, "Directable, high-resolution simulation of fire on the gpu," *ACM Trans. Graph.*, vol. 28, pp. 41:1–41:8, July 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531347

[29] M. Lesieur, O. Mtais, and P. Comte, *Large-Eddy Simulations of Turbulence*. Cambridge University Press, 2005.

[30] M. Wicke, M. Stanton, and A. Treuille, "Modular bases for fluid dynamics," *ACM Trans. Graph.*, vol. 28, pp. 39:1–39:8, July 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531345

[31] N. Huang, Z. Shen, S. Long, M. Wu, H. Shih, Q. Zheng, N. Yen, C. Tung, and H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *PROCEEDINGS OF THE ROYAL SOCIETY OF LONDON SERIES A-MATHEMATICAL PHYSICAL AND ENGINEERING SCIENCES*, vol. 454, no. 1971, pp. 903–995, MAR 8 1998.

[32] N. Huang and S. Shen, *The Hilbert-Huang Transform and Its Applications*. World Scientific Publishing Company, 2005.

[33] J. Nunes, Y. Bouaoune, E. Delechelle, O. Niang, and P. Bunel, "Image analysis by bidimensional empirical mode decomposition," *IMAGE AND VISION COMPUTING*, vol. 21, no. 12, pp. 1019–1026, NOV 1 2003.

[34] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decomposition based on local extrema," *ACM Trans. Graph.*, vol. 28, pp. 147:1–147:9, December 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618493

[35] C. Damerval, S. Meignen, and V. Perrier, "A fast algorithm for bidimensional EMD," *IEEE SIGNAL PROCESSING LETTERS*, vol. 12, no. 10, pp. 701–704, OCT 2005.

[36] Z. Liu and S. Peng, "Boundary Processing of bidimensional EMD using texture synthesis," *IEEE Signal Processing Letters*, vol. 12, no. 1, pp. 33–6, January 2005.

[37] P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola, "Interactive sph simulation and rendering on the gpu," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, pp. 55–64. [Online]. Available: http://dl.acm.org/citation.cfm?id=1921427.1921437

[38] L. D. Landau and E. Lifshitz, *Fluid Mechanics, Second Edition: Volume 6 (Course of Theoretical Physics)*. Butterworth-Heinemann, 1987.

[39] M. Pharr and G. Humphreys, *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann, August 2004.