EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

A Fast Simulation Method Using Overlapping Grids for Interactions between Smoke and Rigid Objects

Yoshinori Dobashi[1] Yasuhiro Matsuda[2] Tsuyoshi Yamamoto[1] Tomoyuki Nishita[2]

[1]Hokkaido University, Sapporo, Japan
[2]The University of Tokyo, Tokyo, Japan
{doba, yamamoto}@nis-ei.eng.hokudai.ac.jp, mazda, nis@is.s.u-tokyo.ac.jp

Abstract

Recently, many techniques using computational fluid dynamics have been proposed for the simulation of natural phenomena such as smoke and fire. Traditionally, a single grid is used for computing the motion of fluids. When an object interacts with a fluid, the resolution of the grid must be sufficiently high because the shape of the object is represented by a shape sampled at the grid points. This increases the number of grid points that are required, and hence the computational cost is increased. To address this problem, we propose a method using multiple grids that overlap with each other. In addition to a large single grid (a global grid) that covers the whole of the simulation space, separate grids (local grids) are generated that surround each object. The resolution of a local grid is higher than that of the global grid. The local grids move according to the motion of the objects. Therefore, the process of resampling the shape of the object is unnecessary when the object moves. To accelerate the computation, appropriate resolutions are adaptively-determined for the local grids according to their distance from the viewpoint. Furthermore, since we use regular (orthogonal) lattices for the grids, the method is suitable for GPU implementation. This realizes the real-time simulation of interactions between objects and smoke.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.6.8 [Simulation and Modeling]: Types of Simulation - Animation; I.6.3 [Simulation and Modeling]: Applications.

1. Introduction

Recently, many researchers have focused on the visual simulation of natural phenomena related to fluids. The common methods for simulating fluids can be roughly classified into two generic techniques: the grid-based approach and the particle-based approach. The grid-based approach computes the motion of the fluid by generating a grid in the simulation space [FM97] [Sta99]. Navier-Stokes equations (denoted as 'NS equations' in the following) are discretized by using a grid, and a finite difference method is often used. On the other hand, the particle-based approach represents the fluid media as a set of particles [KTO95] [MCG03] [PTB*03]. Many particles are generated in the simulation space in order to solve the NS equations numerically. Hybrid methods that combine the grid-based and the particle-based approaches have also been proposed [ZB05]. These methods can realize the realistic animation of fluids.

In this paper, we focus on an efficient simulation of smoke interacting with rigid objects. Although efficient particle-based methods for the interaction between rigid objects and fluids have been proposed recently [CBP05] [PK05] [APKG07], we focus on a grid-based approach in this paper. Discussions regarding the rival merits of the particle-based and the grid-based approaches are beyond the scope of this paper. The most significant advantage of the grid-based approach is its simplicity in terms of both its implementation and the algorithm that is employed. Furthermore, when we use regular lattices, significant acceleration of the computation is realized by using graphics processing units (GPUs). Another aspect of the grid-based approach is that the computational cost increases in proportion to the number of grid points. When an object interacts with smoke, the grid resolution must be high enough to represent the shape of the object. Therefore, the number of grid points increases for objects with complex shapes, resulting in higher computa-

tional costs. To address this problem, adaptive methods that generate grids according to the shape of the object have been proposed [LGF04] [FOK05]. However, the grid generation process has to be repeated every time the object moves. GPU implementations of these methods are difficult due to the uneven shapes of the grids. The use of adaptive methods therefore increases the level of accuracy, but, at the same time, the advantages of the grid-based approach are sacrificed.

To address the problems mentioned above, we propose a method that uses multiple overlapping grids. In addition to the grid covering the whole simulation space, another grid is generated separately for each object. We call the grid for the whole simulation space the *global grid* and the grids surrounding the objects are *local grids*. The global grid and the local grids overlap with each other. The local grids translate and rotate in accordance with the motion of the objects. Fluid simulations are carried out separately for each grid, while relevant information is transferred between them. In addition, a set of local grids with various resolutions is assigned to each object for further acceleration. We use a grid with a lower resolution when the object is far from the viewpoint, since the small-scale features of the flow are not visible. This reduces the number of grid points and therefore reduces the computation time. Our method has the following advantages.

- Implementation is easy. The fluid solver [Sta99] that is commonly used can directly solve the fluid motion in each grid.
- No resampling processes are needed for moving objects because the relative positions of the objects and the local grids remain static. This not only saves computation time but also makes it easy to handle moving objects on the GPU. The artifacts that are caused by resampling the shapes of objects for each time-step are also reduced.
- A significant acceleration is possible by using the GPU. Since we use regular (orthogonal) lattices for each grid, the computation is easily carried out on the GPU.

The rest of the paper is organized as follows. Section 2 discusses the previous grid-based methods. The basic concepts behind our method are explained in Section 3, while Section 4 describes the details of our method. Section 5 explains a method for determining the resolution of the local grids. The technique that we use for the GPU acceleration is described in Section 6. In Section 7, several examples are demonstrated and the usefulness of our method is discussed. The limitations of our method are discussed in Section 8. Finally, Section 9 concludes the paper.

2. Related Work

Since our method is a grid-based approach, we should first discuss the grid-based methods that have been developed previously.

Fluid simulation is often achieved by solving the NS equations numerically [KH84] [YU86] [FM97]. After the Stam's paper [Sta99] which solved the problem of stability for numerical simulations, a wide variety of methods have been developed. Examples include simulations of fire and water using [FF01] [NFJ02], simulations of suspended particle explosions [FOA03], and an efficient method for simulating interactions between rigid objects and fluids [CMT04]. Since these methods use a single grid, the number of grid points must increase in order to accurately represent the shape of the objects that are interacting with the fluids.

Adaptive mesh refinement algorithms are often used to accurately represen the shape of the objects [Pop03] [Kho98] [BS99]. Losasso et al. used an octree data structure for the simulation of water and smoke [LGF04]. Guendelman et al. also used the octree data structure to simulate interactions between fluids and thin solids shells [GSLF05]. To further increase the accuracy, methods using tetrahedral meshes have also been proposed [FOK05] [KFCO06]. By using these methods, the interactions between objects and fluids can be calculated with greater accuracy. However, restructuring of the grid/mesh is required when the objects move. The computational algorithm also becomes complex due to the uneven (irregular) structure of the grid. GPU acceleration of these methods is also difficult due to the uneven structure.

Our method addresses these problems by using multiple overlapping grids. We based our method on the techniques developed in the field of computational fluid dynamics (CFD) [BO84] [Fuj95] [Aok01]. The basic algorithm of our method is similar to those described in the CFD literature. However, most of the methods used in CFD assume that there are only one or two moving objects and that the motion of the objects is relatively simple. We have extended the method such that it can handle many rigid objects that are freely experiencing complex motion. The problem in this case is how to transfer the data between multiple overlapping grids with different resolutions. We developed a data transfer method that is suitable for this situation. Furthermore, our method increases the efficiency by applying the idea of LOD (level of detail) techniques. A new method using GPU has also been developed to accelerate the computation.

Shah et al. proposed a method that also uses a grid that moves according to the motion of the fluids [SCP*04]. However, their method uses a single grid and they do not take into account the situation where the object interacts with the fluids. Patel et al. proposed a method using multiple grids [PCCP05]. According to the examples shown in their demonstration movies, this method does not seem to handle the overlapping of the grids. Their multiple grids seem to merge into a single grid when they become close. In this case, we guess that all of the grids need to have a similar resolution in order to merge the grids without losing any information. However, no details of their method are provided, so we have not included further discussion of it here. Treuille et al. proposed an efficient solution to the simulation of interactions between rigid objects and fluids [TLP06]. Their

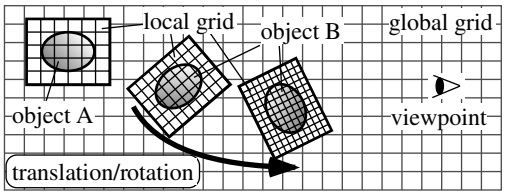


Figure 1: *Basic concept of our method. In addition to a global grid for the whole of the simulation space, a local grid is generated for each object. The local grid moves according to the movement of the object. Its resolution is changed according to its distance from the viewpoint (see object B).*

method also uses multiple overlapping grids, and the motion of the fluids is calculated by projecting a velocity field onto a sub-space represented by a small set of basis fields. The velocity field is represented by a combination of the basis fields. This method significantly accelerates the computation and achieves real-time performance. However, the motions of the flows that can be expressed by this method are affected by the number of basis fields. Furthermore, the memory capacity required for the basis fields is large, and this limits the number of objects that can be used in the simulation. Compared to their method, our method can be considered to be a simulation method that uses multiple grids in full-space, rather than in sub-space. Batty et al. proposed an accurate method for simulating the interactions between fluids and solid objects [BBB07]. This method can simulate accurate fluid motion even on a coarse single regular grid. However, since the shape of the object shape is sampled at the grid points, the grid interval has to be small enough to capture the overall shape in detail. We believe that combining our method with theirs could drastically increase the efficiency of the simulation. We will leave this issue for a future work.

3. Basic Concepts of Our Method

As shown in Fig. 1, we prepare a global grid which covers the whole simulation space and also multiple local grids. Without loss of generality, we assume that the local grids always overlap with the global grid. Our method can handle the overlapping of the local grids when they are in close proximity. The NS equations are solved separately for each grid, but the information for each grid is transferred to each of the other grids at every time-step of the simulation.

Although the flow around an object is complex, the small-scale features of the flow are not visible when the object is far from the viewpoint, so we make use of this fact to reduce the computational cost. We prepare a set of local grids with different resolutions for each object. During the simulation, the appropriate resolution of each grid is determined based on its distance from the viewpoint (see object B in Fig. 1). In addition, a specific local grid is generated when smoke exists near the viewpoint.

Each of the grid-points in the local grids is assigned a status that indicates whether the grid-point is inside the object or not. This discretization process is carried out only once in a preprocess stage, since the local grids move according to the motion of the object. When there is a local grid surrounding the viewpoint, it moves according to the motion of the viewpoint.

For the fluid simulation of the local grids, we assume that the objects are static. Instead, inertial forces are added as external forces in order to take the motion of the local grids into account.

4. Smoke Simulation Using Overlapping Grids

First, we explain the governing equations in Section 4.1. Next, the computational procedure is explained in Section 4.2. The method for transferring information between grids is explained in Section 4.3.

4.1. Governing Equations

The motion of smoke is calculated by the numerical analysis of the following incompressible NS equations.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f}, \quad (1)$$

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

where $\mathbf{u}$ is the velocity, t is the time, ρ is the density, p is the pressure, and $\mathbf{f}$ is the external force. These equations are solved numerically by using the global grid and the local grids. In the case of local grids, the external force includes the inertial force due to the movement of the local grids. The inertial force works at each grid point and is expressed by:

$$\mathbf{f}_{inertia} = -\frac{d^2\mathbf{c}}{dt^2} - \frac{d\Omega}{dt}\times\mathbf{x} - 2\Omega\times\mathbf{u} - |\Omega|^2\mathbf{x}, \quad (3)$$

where $\mathbf{c}$ is the center position of the local grid, Ω is the angular velocity of the local grid, $\mathbf{x}$ is the position of each grid point, and $\mathbf{u}$ is the velocity at each grid point. The terms on the right-hand side indicate the inertial forces due to the acceleration of the local grid, the force due to the angular acceleration, the Coriolis force, and the centrifugal force, respectively. The NS equations are solved in the following way [CMT04]. First, the approximate solution $\tilde{\mathbf{u}}$ is calculated by ignoring the pressure term $-\frac{1}{\rho}\nabla p$ in Eq. (1).

$$\tilde{\mathbf{u}} \approx \mathbf{u}(t) + \Delta t(-(\mathbf{u}(t)\cdot\nabla)\mathbf{u}(t) + \nu\nabla^2\mathbf{u}(t) + \mathbf{f}), \quad (4)$$

where Δt is the time-step of the simulation and $\mathbf{u}(t)$ is the velocity at time t. Next, the velocity $\mathbf{u}$ at time $(t+\Delta t)$ is obtained by taking into account the pressure term, that is,

$$\mathbf{u}(t+\Delta t) = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho}\nabla p. \quad (5)$$

To compute the above equation, the pressure p is required. Therefore, we put the above equation into Eq. (2).

$$\frac{\Delta t}{\rho}\nabla^2 p = \nabla\cdot\tilde{\mathbf{u}}. \quad (6)$$

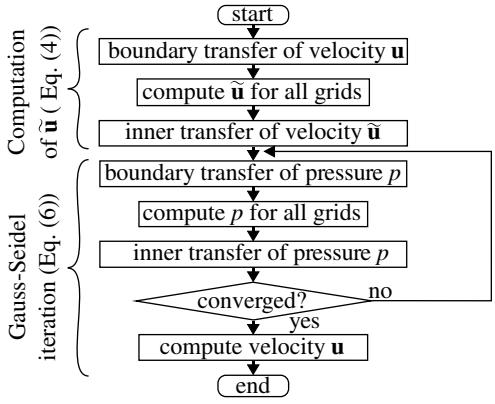


Figure 2: *Computational procedure for each time-step.*

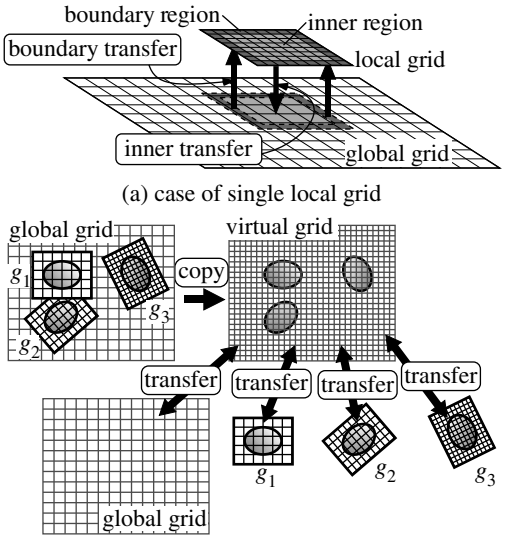
The pressure p is obtained by solving Eq. (6). Then Eq. (5) is calculated to obtain $\mathbf{u}(t+\Delta t)$. Eq. (6) can be solved by using an iterative technique such as the Gauss-Seidel method, which we use for our calculations.

4.2. Numerical Simulation

Fig. 2 shows the computation flow that is executed in each time-step. Before describing the details of the simulation process, let us define *a boundary region* and *an inner region* for each grid (see Fig. 3(a)). The boundary region consists of several grid points near the boundary of the local grid. The inner region is the region inside the boundary region. Correspondingly, there are two types of information transfer between the grids: *a boundary transfer* and *an inner transfer*. In the case of the boundary transfer, physical values such as velocities and pressures are interpolated from the global grid. However, if another local grid overlaps at the boundary region and its resolution is lower, the values on the boundary region are interpolated from the lower resolution grid. The inner transfer replaces the physical values of the global grid by the values in the inner region of the local grid. If another local grid overlaps and its resolution is lower, the values of the lower resolution grid are also replaced by the values of the higher resolution grid.

The width of the boundary region significantly affects the accuracy of the simulation. In Fig. 3(a), the boundary region includes the outermost three grid points of the local grid. Theoretically, the outermost two grid points are sufficient in order to avoid discontinuities between the local grids and the global grid, since the NS equations include a second derivative term. However, we include the outermost four grid points according to our experiments shown in Section 4.3.3.

Now, let us explain the numerical simulation process by using Fig. 2. First, the boundary transfer procedure is carried out for the velocities. That is, the velocities on the boundary region of each local grid are interpolated from the global grid or a lower resolution grid overlapping at the boundary region. Using these velocities as boundary conditions, the approximate velocity $\tilde{\mathbf{u}}$ (Eq. (4)) is calculated for all grids. After that, the inner transfer is carried out for each grid. Next, the pressure is calculated iteratively by using the Gauss-Seidel method. At each iteration, the pressure is transferred among the local grids and the global grid following the same process as used for the velocities (see Fig. 2). Transferring the pressure at each iteration eliminates the differences in the solutions between grids. After the iteration has converged, the velocities are updated by using Eq. (5). These processes are repeated for each time-step to calculate the fluid flows.



(a) case of single local grid

(b) basic idea for case of multiple local grids

Figure 3: *Transfer processes between grids.*

For local grids, we use fixed boundary conditions when solving the NS equations. For the global grid, we use periodic or fixed boundary conditions [Sta99].

4.3. Transfers of Physical Values between Grids

First, we explain the algorithm by using a simple case where there is a single local grid (Section 4.3.1). Next, the transfer processes are explained for a general case where multiple local grids exist (Section 4.3.2). In Section 4.3.3, we verify the validity of our transfer method by showing an experimental result.

4.3.1. Case of Single Local Grid

Fig. 3(a) shows the idea of the transfer process for the simple case. As mentioned before, in the boundary transfer process, the velocities and the pressures at the boundary region of the local grid are overwritten by the corresponding velocities and pressures of the global grid. Next, the inner transfer process replaces the velocities and the pressures of the global grid by those of the inner region of the local grid. In these

```
BoundaryTransfer() {
  for(i = 0; i < n; i++) {
    for(j = 1; j < n; j++) {
      if(i != j && g_i overlaps g_j) {
        copy velocities/pressures of g_i to boundary region of g_j;
        copy velocities/pressures of object of g_i to g_j;
      }
    }
  }
}
```

(a) boundary transfer

```
InnerTransfer() {
  for(i = 1; i < n; i++) {
    for(j = 0; j < i; j++) {
      if(g_i overlaps g_j) {
        copy velocities/pressures at inner region of g_i to g_j;
      }
    }
  }
}
```

(b) Inner transfer

Figure 4: *Pseudo codes for transfer processes.*

processes, the velocities/pressures at arbitrary positions are linearly interpolated from the values stored at the grid points. The velocities have to be transformed between the local coordinate system associated with the local grid and the global coordinate system. We assume that the origin of the local coordinate system is at the center position $\mathbf{c}$ of the local grid. The velocity $\mathbf{u}_l$ at position $\mathbf{x}_l$ in the local coordinate system is transformed into the velocity $\mathbf{u}_g$ in the global coordinate system by the following equation.

$$\mathbf{u}_g = \mathbf{R}^{-1}\mathbf{u}_l + \frac{d\mathbf{c}}{dt} + \Omega \times (\mathbf{R}^{-1}\mathbf{x}_l), \tag{7}$$

where matrix $\mathbf{R}$ indicates the rotation of the local coordinate system. The second and the third terms in the right-hand side of Eq. (7) are associated with the translation and the rotation of the local grid, respectively. The transform from $\mathbf{u}_g$ at position $\mathbf{x}_g$ in the global coordinate to $\mathbf{u}_l$ is expressed by:

$$\mathbf{u}_l = \mathbf{R}(\mathbf{u}_g - \frac{d\mathbf{c}}{dt} - \Omega \times (\mathbf{x}_g - \mathbf{c})). \tag{8}$$

4.3.2. Case of Multiple Local Grids

As shown in Fig. 3(b), the basic concept for the case of multiple local grids is to use a virtual grid that has the highest resolution. First, the physical values (i.e. velocities, pressures and densities of smoke) of all of the grids are copied into this virtual grid (see Fig. 3(b)). This copy process is carried out in ascending order in terms of grid resolutions. The physical values in the grids with higher resolutions overwrite the values in the grids with lower resolutions. At those grid points in the virtual grid which correspond to objects, the velocities and the pressures of the objects are written. We assume the pressures inside the objects to be zero. Then the method described in Section 4.3.1 is used for each grid by assuming the virtual grid as the global grid. Note that no fluid simulations are carried out on the virtual grid. The virtual grid is just a temporal buffer for the transfers. This method is simple and easy to implement. However this method requires a large amount of memory capacity for the virtual grid. Therefore, we have developed an alternative method that can obtain the same result without using the virtual grid. The details are described below.

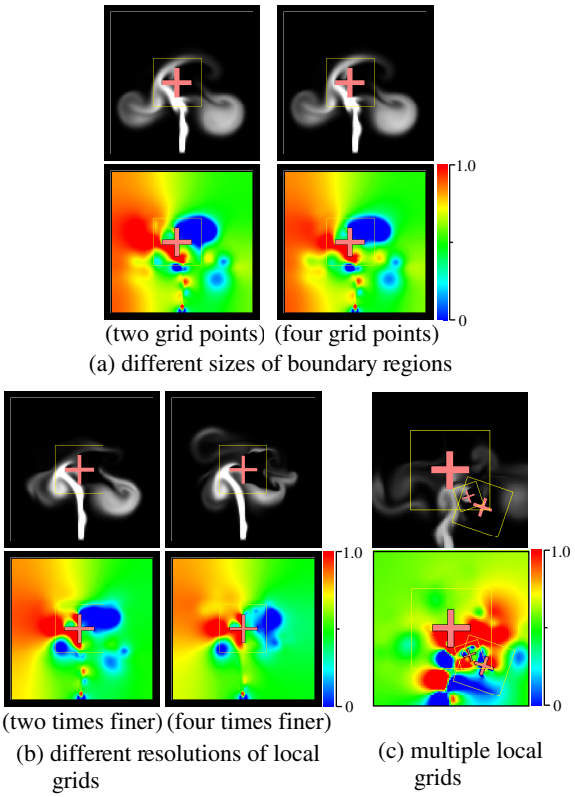
First, all of the grids, including the global grid, are sorted into ascending order in term of grid resolution. In the following, let us denote each grid as $g_i (i = 1, 2, \cdots, n)$ after the sorting process, where n is the number of grids. The boundary transfer for g_j is carried out as follows (see Fig. 4(a)). In the order of $i = 1, 2, \cdots, n$, our method checks whether g_i and g_j overlap or not. If they overlap, the physical values at the boundary region of g_j are replaced and interpolated from the corresponding values of g_i. The pressures and velocities at the grid points corresponding to the objects of g_i are also copied to the corresponding grid points of g_j. Next, the inner transfer for g_j is as follows (see Fig. 4(b)). For a grid g_i that is finer than g_j (i.e., $i > j$), our method checks whether g_i and g_j overlap or not. If they overlap, the physical values of grid g_j are replaced by those of g_i.

In the above processes, the velocities are transformed between the local coordinates of local grids g_i and g_j. This transformation is carried out by means of the global coordinate system. The velocities of g_i are transformed into the global coordinate system by using Eq. (7) and are then transformed into the local coordinate of g_j by using Eq. (8).

4.3.3. Verification of Our Transfer Method

We performed three experiments to verify the validity of our transfer method. These consist of two-dimensional simulations of interactions between a cross-shaped object and smoke. The resulting images are summarised in Fig. 5. Density and pressure distributions are shown. Pressures are converted into pseudo colors after normalizing them so that the maximum pressure corresponds to 1.0. Please refer to the accompanying movie files for animations of these experiments.

First, Fig. 5(a) shows the effect of the number of grid points for the transfer of the boundary region of the local grids. The resolution of the local grid is the same as that of the global grid. The results when two or four outmost grid points are used are shown. When using the outermost two grid points, artifacts are observed at the boundary of the local grid. The artifacts disappeared when the four outermost grid points were used. However, this number is dependant on how the time-step of the simulation satisfies the Courant-Friedrichs-Lewy condition (or CFL condition) [FM97]. The CFL condition indicates that the time-step should be smaller than (grid interval)/(maximum speed of the flow). In this experiment, we verified that the time-step was 1.5 times greater than the CFL condition. When the user specifies a larger time-step, the flow outside the local grid may flow across the boundary region during a single time-step. In this case,



(a) different sizes of boundary regions

(b) different resolutions of local grids

(c) multiple local grids

Figure 5: *Verification of transfer method. Density and pressure distributions are shown. Pressures are converted into pseudo colors. The yellow rectangles in these images indicate the region covered by local grids. Note that (c) shows magnified images in order to observe the details.*

the number of grid points for the boundary region needs to be larger.

Next, we investigated the effect of differences in resolution between the local and the global grids. Fig. 5(b) shows the results for local grids with two and four times finer resolutions than that of the global grid. For the two times finer resolution, no discontinuities between the local and the global grids are noticeable. Although slight discontinuities are observed in the pressure distribution when using the four times finer resolution, it is hardly distinguishable in terms of the density distribution. When using eight times finer resolution, we observed discontinuities, because the CFL condition is highly violated. The artifacts can be avoided by using a shorter time-step although the computation time becomes longer. In this paper, we assume that the ratio of the resolutions between the grids is less than four.

We then verified the transfer of data between multiple grids. Fig. 5(c) shows the results using three cross-shaped objects of different sizes. The largest object does not move and its local grid has the same resolution as that of the global grid. The other two objects rotate about themselves while

global grids	local grids	total times	times for transfers
130x130	50x50	13	2.5 (19%)
130x130	100x100	19	4.7 (25%)
130x130	200x200	43	13 (28%)
130x130	50x50, 60x60, 60x60	23	8.9 (39%)

Table 1: *Computation times in milliseconds for a single time step. The ratio of the time required for data transfer to the total time are shown in brackets in the last column.*

revolving around the largest object. The resolutions of the local grids for the middle-sized object and the smallest object are respectively two and four times finer than that of the global grid. Note that the images in Fig. 5(c) have been magnified in order to observe the details. As shown in these images, there are few artifacts and continuous distributions are obtained. However, in the accompanying movie, we can observe the presence of some artifacts when the smallest object passes under the largest object. This is because the CFL condition is not satisfied due to the faster velocity of the flow around this region. In this instance, the time-step is 6 times larger than that of the CFL condition; that is, the CFL condition is completely violated. To prevent this, it is advisable to choose the time-step such that the CFL condition is satisfied for the grid with the highest resolution. When the CFL condition is satisfied for the highest resolution grid, it is always satisfied for all of the other grids. This indicates that fluids from outside a local grid do not flow into the inner region during a single time-step. This also provides an additional benefit in that we can compute the advection process separately for each grid.

The computation times for a single time step are shown in Table 1. The data transfer processes consume 20 - 40 % of the total computation time. When multiple local grids are used, the computation time for the data transfer does not increase proportionally to the number of the grids (see the fourth row in the table). This is because the data transfer processes are not carried out for non-overlapping grids. Compared to the use of an adaptive approach [KFCO06], the ratio of the data transfer in our method is comparable to the ratio used for the remeshing.

5. Determination of Resolutions of Local Grids

As described in Section 3, a set of local grids with different resolutions are assigned to each object. For an object far from the viewpoint, we use a grid with a relatively-lower resolution. When the object is nearer, we basically use a grid with a higher resolution. However, if the flow around the object is relatively simple, we can still use a coarser grid. Therefore, we determine an appropriate resolution, taking into account both the complexity of the flow and the distance from the viewpoint. However, it is not an easy task to measure the complexity of the flow. We heuristically use the Reynolds number, *Re*, as a measure of the complexity.

The Reynolds number is the ratio of inertial forces to viscous forces and is often used to identify the type of a flow, from a laminar flow to a turbulent flow. A larger Reynolds number corresponds to a more turbulent flow. We consider turbulent flow to be a complex flow and determine the resolution of the local grid in proportion to the Reynolds number. The Reynolds number is expressed as $Re = UL/\nu$, where U is the mean speed of the flow, L is the characteristic length, and ν is the kinematic fluid viscosity.

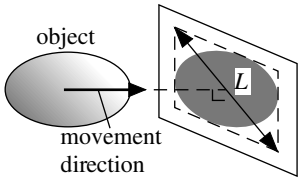


Figure 6: *Computation of characteristic length.*

We assume that m local grids are prepared for each object, and that the grid-interval between the grid points of grid g_k is $h_k (k = 1, 2, \cdots, m)$. The distance d from the viewpoint to the center position of the object and the Reynolds number Re are calculated at each time-step. In computing Re, we use the sum of the translation speed and the rotation speed of the object as the mean speed U of the flow. The characteristic length L is determined as follows (see Fig. 6). The object is projected onto a plane perpendicular to the direction of movement of the translation, and a bounding box is calculated on the projected plane. We then take the length of the diagonal of the bounding box as the characteristic length. Next, using the distance d and the Reynolds number Re, we select a local grid with the maximum grid interval that satisfies the following condition.

$$\frac{h_k}{d} Re < \varepsilon, \tag{9}$$

where ε is a user-specified threshold. Let us now discuss the meaning of the above equation. When the grid is projected on the screen, the grid-interval h_k corresponds to $h_k \frac{e}{d}$ pixels. e is a constant determined by camera parameters, such as the viewing angle. It is sufficient to choose h_k so that $h_k \frac{e}{d} = 1$ (or $h_k = \frac{d}{e}$) since any features of the flow that are smaller than one pixel are not visible. Now, let us define a function, $A(Re)$, of the Reynolds number that indicates the average size of the vortices in the flow. We consider that the flow features are captured by reproducing the representative vortices in the flow. Therefore, when the size $A(Re)$ is large, a longer grid-interval can be used to reproduce the vortices. However, the grid-interval should not exceed $A(Re)$. Taking into account the above discussion, we choose the maximum grid-interval that satisfies $h_k < \frac{d}{e} A(Re)$. Let us now discuss the function $A(Re)$. As the flow becomes more turbulent, i.e., the Reynolds number increases, many small vortices appear, and so we can assume that $A(Re)$ decreases as the Reynolds number increases. One of the simplest functions that satisfies this property is $A(Re) = \alpha/Re$, where α is a constant. Then, the condition for h_k becomes $h_k < \frac{d}{e} \frac{\alpha}{Re}$. When we let $\varepsilon = \alpha/e$, then Eq.(9) is obtained. Values for ε have to be chosen by taking into account the size of the vortices. However, we determine ε experimentally in our current implementation. In the examples shown in Section 7, ε is set to 1.5. A method for the automatic determination of ε (or α) will be part of our future work.

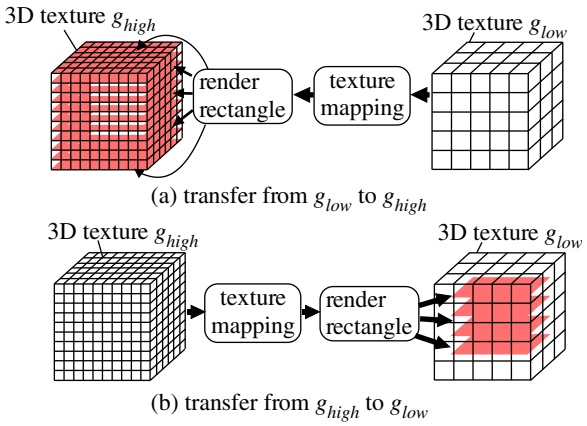


Figure 7: *Data transfer between two grids on GPU.*

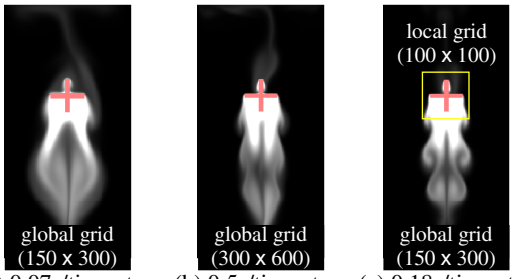
6. GPU Acceleration

An important advantage of our method is that it is suitable for the implementation of GPU acceleration. Since the grids are represented by regular lattices, physical quantities such as velocities and pressures can be stored directly in two- or three-dimensional textures. We use the method proposed by Crane et al. to solve the NS equations on the GPU [CLT07]. However, in this method, the shapes of the objects have to be resampled every time they move. In our method, we prepare an object texture in a preprocessing stage and the texture is sent to the GPU only once. At each texel of the object texture, a flag indicating whether the corresponding grid point is inside the object or not is stored. This texture need not be updated throughout the simulation.

Next, we will explain a method for data transfer between grids on the GPU. As shown in Fig. 7, let us explain the data transfer between two 3D grids, a low resolution grid g_{low} and a high resolution grid g_{high}. The data transfer processes are realized by using a function called `render-to-texture`. For the boundary transfer, we render rectangles corresponding to the boundary region of grid g_{high} at each slice of the 3D texture (Fig. 7(a)). The textures of g_{low} are mapped onto these rectangles. Next, for the inner transfer, we render a set of rectangles at the inner region of g_{low} (Fig. 7(b)). The textures of g_{high} are mapped onto the rectangles.

7. Results

The following examples are calculated by using a desktop PC with a Pentium 4 3.8GHz processor and 2.0 GB of main memory. For the GPU acceleration, we use nVidia GeForce 8800 Ultra. Please refer to the accompanying movie files for animations of the examples shown in this section.



(a) 0.07s/time step (b) 0.5s/time step (c) 0.18s/time step

Figure 8: *Comparison of two-dimensional simulations with different grid configurations.*

7.1. Experimental Results

In this subsection, we implement our method on a CPU and we verify the validity of our method using simple examples.

First, we verify the effectiveness of our method by using the two-dimensional simulations shown in Fig. 8. A two-dimensional cross-shaped object that is moving up and down is shown. Figs. 8(a) and (b) use only global grids. Fig. 8(c) uses both a global grid and a local grid. We consider Fig. 8(b) as a reference solution, since this grid has the highest resolution. The average computation time spent on each time-step is shown in the captions. The computation for Fig. 8(a) is fast, but small-scale vortices have disappeared. Our method reproduces the small-scale features (see Fig. 8(c)), and is almost three times faster than that for Fig. 8(b).

Next, we verify the effectiveness of our method for determining the resolution of the local grid. We use a simple three-dimensional simulation, in which a fighter plane rotates and descends, as shown in Fig. 9(a). We compare two types of simulations. In Fig. 9(b), a global grid and a local grid are used, and the resolution of the local grid is fixed. This simulation is used to obtain a reference solution. In Fig. 9(c), an appropriate grid is determined from three grids with different resolutions for the fighter plane. Figs. 9(d) and (e) show the results, which correspond to Figs. 9(b) and (c) respectively. The average computation times are shown in the captions. Our method can generate images similar to the reference solution, yet the computation process is 8 times faster.

7.2. Examples

Fig. 10 shows a real-time two-dimensional simulation of interactions between several objects and rising smoke. The shapes of the objects are determined by bitmap images of the letters that make up the word 'EUROGRAPHICS'. Each letter is represented by a local grid with 128^2 grid points. The number of grid points for the global grid is 512^2. The complex motions of smoke due to the interactions with the objects have been simulated. This simulation was carried out on the GPU and the simulation achieved 28 fps (frames per second).

Fig. 11 shows an example of an interactive operation. The user controls a fighter plane that has been placed in a fluid. The numbers of grid points of the global and the local grids are 128^3 and $41 \times 59 \times 25$, respectively. This example was calculated on the GPU and the simulation achieved a frame rate of 10 fps. To investigate the degree of acceleration achieved by GPU, we computed the same simulation on the CPU. The frame rate in this case was 0.14 fps. That is, 71 times faster simulation was achieved when using the GPU.

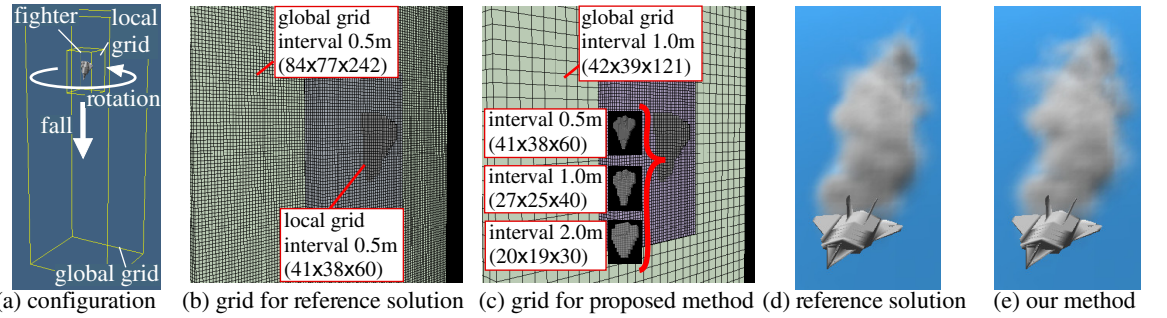
The final example is a simulation of the dynamic smoke that has been generated from 15 fighter planes that are under attack by missiles, as shown in Fig. 12. Fig.12(a) shows the grids that were used for this simulation. For each fighter, three local grids with different resolutions are assigned and an appropriate grid is determined. Another local grid is also generated surrounding the viewpoint. This example is calculated on the CPU. The average computation time per time-step is 10 seconds. Fig. 12(b) shows an image from the animation. Realistic motion of smoke can be simulated.

8. Discussion

When using our method, appropriate resolutions of the local and the global grids need to be chosen in order to compute the complex flow due to interactions between the objects and smoke. The resolutions should be chosen such that the shapes of the objects are represented with sufficient accuracy after they are sampled at the grid points. Although it might be possible to determine the resolutions automatically, we determine them experimentally in our current implementation. As we described before, our method cannot eliminate the artifacts that occur due to the voxelization. The solution to this problem is to combine our method with the variational framework proposed by Batty et al [BBB07].

The size of the local grid (i.e., the region covered by the local grid) is also important. When the size is too small, the flow around the objects can not be calculated accurately, so we determined the size by trial and error. We examined the flow around the object in advance by using local grids with different sizes. We then chose the size that adequately reproduced the flow around the object. Automatic determination of an appropriate size will be part of our future work.

One drawback of our method is that the memory consumption becomes large when there are many moving objects. Compared to the adaptive approaches [LGF04] [FOK05], our method may require a larger amount of memory. Although the adaptive approaches require extra memory to store the complex data structures, they only subdivide the grid or mesh at the boundaries of the objects. In addition, when objects overlap, the adaptive approaches subdivide the boundaries of the set of objects, not the boundaries of each object. In contrast, our method always uses regular grids around the objects. However, we believe that this is not a problem, since the memory capacity of CPU/GPU has increased dramatically in recent years.



(a) configuration (b) grid for reference solution (c) grid for proposed method (d) reference solution (4.6s/time step) (e) our method (0.56s/time step)

Figure 9: *Comparison for resolution determination of local grid.*

Methods for deformable objects will also form part of our future work. When an object is deformed, its shape has to be resampled at the grid points of the local grid. One solution could be to deform the grid according to the deformation of the object. In this case, inertial forces due to the deformation have to be taken into account. An extended differential operator using the deformed grid is also required. However, it is difficult to treat highly deformed objects such as cloth.

Finally, our method depends in part on the use of the Gauss-Seidel method to solve the mass conservation equation (Eq. 2). Use of the Gauss-Seidel method provides a chance to transfer pressure during the iteration process and this results in the continuous solution between the grids. However, one problem that brings is that the convergence speed is slow. The use of a faster method, such as preconditioning conjugate gradient method, will be investigated.

9. Conclusion

In this paper, we have proposed a method using multiple overlapping grids for the efficient simulation of interactions between smoke and rigid objects. A local grid is generated for each object, and the grids move according to the motion of the objects. This makes it easy to handle dynamic objects in the simulation. A method for determining the resolutions of the local grids that enables further acceleration has also been proposed. We have demonstrated that our method can achieve significant speeding up of the simulation process. We have also proposed a technique for the further acceleration of our method by using a GPU.

Acknowledgement

This research is partly supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) from the Ministry of Internal Affairs and Communications of Japan.

Figure 10: *Real-time two-dimensional simulation of interactions between objects and rising smoke.*

Figure 11: *Interactive simulation. The user can move the fighter plane.*

References

[Aok01] AOKI T.: 3d simulation for falling papers. *Computer Physics Communications 142* (2001), 326–329. 2

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. on Graphics 26*, 3 (2007), 457–462. 1

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. on Graphics 26*, 3 (2007), 100. 3, 8

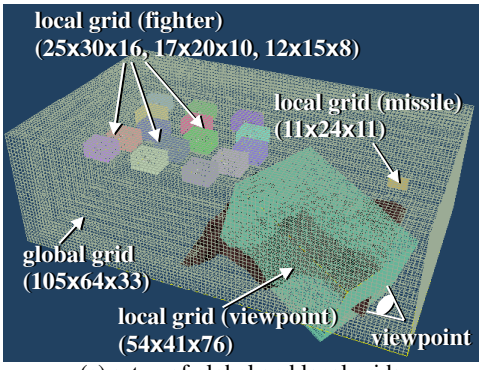
[BO84] BERGER M. J., OLIGER J.: Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Comp. Phys. 53*, 3 (1984), 484–512. 2

[BS99] BENNETT B. A. V., SMOOKE M. D.: Local rectangular refinement with application to nonreacting and reacting fluid flow problems. *Journal of Comp. Phys. 151*, 2 (1999), 684–727. 2

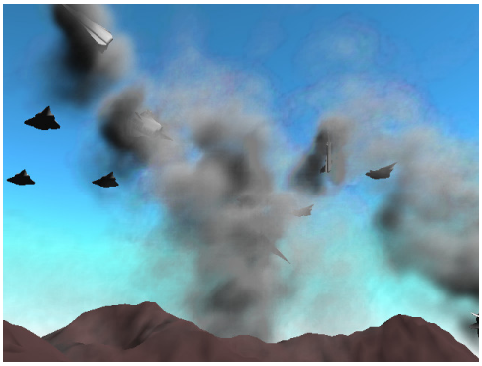
[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005* (2005), pp. 219–228. 1

[CLT07] CRANE K., LLAMAS I., TARIQ S.: Real-time simulation and rendering of 3d fluids. In *GPU Gems 3* (2007), pp. 633–675. 7

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. on Graphics 23*, 3 (2004), 377–384. 2, 3



(a) setup of global and local grids



(b) image at time t = 10s

Figure 12: *Simulation of dynamic smoke generated from fighters.*

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. SIGGRAPH 2001* (2001), pp. 23–30. 2

[FM97] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *Proc. SIGGRAPH 1997* (1997), pp. 181–188. 1, 2, 5

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. *ACM Trans. on Graphics 22*, 3 (2003), 708–715. 2

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. *ACM Trans. on Graphics 24*, 3 (2005), 904–909. 2, 2, 8

[Fuj95] FUJII K.: Unified zonal method based on the fortified solution algorithm. *J. Comp. Phys. 18*, 1 (1995), 92–108. 2

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. on Graphics 24*, 3 (2005), 973–981. 2

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. *ACM Trans. on Graphics 25*, 3 (2006), 377–384. 2, 6

[KH84] KAJIYA J. T., HERZEN B. P. V.: Ray tracing volume densities. *Computer Graphics 18*, 3 (1984), 165–174. 2

[Kho98] KHOKHLOV A. M.: Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *Journal of Comp. Phys. 143*, 2 (1998), 519–543. 2

[KTO95] KOSHIZUKA S., TAMAKO H., OKA Y.: A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Dynamics Journal 29*, 4 (1995), 29–46. 1

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. on Graphics 23*, 3 (2004), 457–462. 2, 2, 8

[MCG03] MULLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. Symposium on Computer Animation 2003* (2003), pp. 154–159. 1

[NFJ02] NGUYEN D., FEDKIW R., JANSEN H.: Physically based modeling and animation of fire. *ACM Trans. on Graphics 21*, 3 (2002), 721–728. 2

[PCCP05] PATEL S., CHU A., COHEN J., PIGHIN F.: Fluid simulation via disjoint translating grids. In *SIGGRAPH 2005 Technical Sketch* (2005). 2

[PK05] PARK S., KIM M. J.: Vortex fluid for gaseous phenomena. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005* (2005), pp. 261–270. 1

[Pop03] POPINET S.: Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Comp. Phys. 190*, 2 (2003), 572–600. 2

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 335–343. 1

[SCP*04] SHAH M., COHEN J. M., PATEL S., LEE P., PIGHIN F.: Extended galilean invariance for adaptive fluid simulation. In *Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA) 2004* (2004), pp. 213–221. 2

[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH'99* (1999), pp. 121–128. 1, 2, 2, 4

[TLP06] TREUILLE A., LEWIS A., POPOVIC Z.: Model reduction for real-time fluids. *ACM Trans. on Graphics 25*, 3 (2006), 826–834. 2

[YU86] YAEGER L., UPSON C.: Combining physical and visual simulation, creation of the planet jupiter for the film 2010. *Computer Graphics 20*, 4 (1986), 85–93. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. on Graphics 24*, 3 (2005), 985–972. 1