Vis Comput (2013) 29:883–892
DOI 10.1007/s00371-013-0848-7

ORIGINAL ARTICLE

A novel, integrated smoke simulation design method supporting local projection and guiding control over adaptive grids

Qing Zuo · Yue Qi · Hong Qin

Published online: 5 June 2013

Abstract High-fidelity smoke simulation in a large-scale complex environment is extremely time-consuming due to the expensive computational cost of using highly dense regular grids. There have been quite a few improved algorithms/techniques aiming to enhance the simulation's visual effects and reduce the time consumption during the last two decades. However, most of the state-of-the-art methods will encounter difficulties of not being able to model fine turbulent details during simulation or losing high-frequency shape details at the fine scale when simulated smoke interacting with nearby obstacles. One straightforward solution is to continue to refine spatial resolution at the expense of increased time complexity. This paper, however, advocates an improved strategy for smoke simulation design over adaptive grids, while simultaneously enabling the functionalities of local projection and guiding control. First, our new integrated method supports adaptive grid projection that can significantly reduce the computational cost during the velocity projection phase. During smoke simulation design, the use of adaptive grids flexibly accommodates finer cells near obstacles with fine details, and coarser cells anywhere else, as a result, fine-scale object features can be faithfully retained without the need of global grid refinement. Second, our integrated solution over adaptive grids can tightly couple guiding control with local projection, which is capable of handling tiny obstacles that are impossible to model with global coarse grids alone during simulation preview. Comprehensive experiments have shown that our integrated method has the advantage of generating turbulent phenomena when interacting with small-scale features of obstacles, and at the same time offering the preview mechanism for efficient large-scale smoke simulation design.

Q. Zuo (✉) · Y. Qi
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
e-mail: zuoqing1988@aliyun.com

H. Qin
Computer Science Department, Stony Brook University, Stony Brook, NY 11790, USA

Keywords Smoke simulation · Guiding control · Adaptive grid

1 Introduction and motivation

Visually realistic animation of many complex physical phenomena, such as smoke, fire, and water, could be effectively simulated by using grid-based simulation in graphics. However, large-scale fluid simulation to generate many small-scale details using prior methods [1, 2] is extremely time-consuming due to the high computational cost associated with high resolution for the underlying grids. During the last two decades, there have been many research works exploring new techniques to accelerate the simulation.

Among them, two key ideas are of most relevance to our approach in this paper. The first one is to use a coarser grid to solve the most costly step—velocity projection. Losasso et al. [3] used an octree grid to accelerate the simulation. Their method indeed reduces the projection cost. However, it does not generate a underlying fine grid, so it may be difficult to achieve higher-order interpolation and also the details (such as vorticity) in the coarse cells may be lost or weakened. Lentine et al. [4] improved the projection step by mapping the velocity field to a regular coarse grid to solve a global projection, and then conducted local projection for fine cells in each coarse cell. Their method works well in most cases. However, when the embedded obstacles can not be represented faithfully, the local projection will lead to artifact (e.g., no divergence-free or abnormal larger velocity comparing with surrounding cells). The second one is focusing on using a coarse preview simulation to guide and control the fine simulation [5–8]. Using the guiding control method, one can design and adjust the smoke animation very fast on the coarse preview grid, and the final fine-grid simulation needs to be conducted only once. However, existing guiding control methods may not work well in cases where the obstacles cannot be represented well on the coarse grid, and they cannot be integrated with the fast projection method either [3, 4].

To tackle the above problems, we propose a novel integrated design method for smoke simulation. In our method, two main techniques—guiding control and fast velocity projection—are integrated over an adaptive grid. Using our method, one can benefit from two fronts. To obtain the final fine smoke simulation, one can first design and adjust the preview coarse grid and use the coarse simulation to guide the fine simulation. Comparing with directly computing over the fine grid, this is the first advantage. To conduct the final fine-grid simulation, the simulation cost can be reduced by our fast projection method and this is the second advantage.

The key contributions of this paper are as follows:

– We propose an Integration of Adaptive Grid Projection and Guiding Control (IAGPGC) method. The users can benefit from both two advantages of fast projection and preview design;
– In our IAGPGC method, we develop an improved adaptive grid projection method which can model small-scale features of obstacles embedded in the adaptive grid and enhance their interaction with surrounding fluids precisely while significantly reducing the projection cost;
– We develop a novel guiding control method, which can handle the cases where the obstacles cannot be represented well on the coarse preview grid.

2 Related work

Fluid simulation technologies have been gaining popularity since the end of 1990s [1]. Significantly limited by the computational power, earlier works failed to use large grids to produce highly detailed results. Consequently, many researchers explored different methods to improve the simulation. We briefly review the most related work in the following categories.

Higher-order methods To improve the baseline of simulation on the existing grid, earlier research tried to use higher-order interpolation method [2], or more accurate advection schemes such as BFECC, QUICK, MacMormack, and so on [9–11]. Additionally, a mass and momentum conserved method was proposed by Lentine et al. [12].

Enhanced vorticity Fedkiw et al. [2] introduced a vorticity confinement method to help recover some lost momentum by adding external forces according to the existing vorticities. Selle et al. [13] extended this method further by adding vorticity particles. Chen et al. [14] developed Langevin particle method based on a turbulence energy model with turbulence viscosity.

Irregular grid methods Several researchers adopted irregular grids such as octree grid [3], hybrid grid [15], and triangular mesh for smoke simulation [16, 17]. All of these methods have an advantage of better representation of boundaries and can reasonably distribute computing resources. However, the complicated structures hinder our ability to design robust numerical methods. For example, a higher-order interpolation on the octree grid could be rather laborious and inaccurate.

Coarse grid projection In contrast to solving the global projection on the finer grid, many researchers used only a coarse grid projection. High resolution simulation can be obtained by adding turbulences into the coarse grid [18–20]. Although these methods have exhibited some successes at adding details, but they are nonphysical and produce significantly less realistic results than simply increasing the resolution. Recently, Lentine et al. [4] mapped the velocity field to the coarse regular grid to solve the Poisson equation and then mapped them back to the fine grid for local projection. Although their method can accelerate the projection one or two orders of magnitude, artifacts may occur in case that the high frequency obstacles cannot be represented well on the regular coarse grid.

Guiding control Guiding control is a method that the fine resolution simulation is guided by the coarse resolution one and the two shall be matched to judge the guiding quality. Earlier, Treuille et al. achieved keyframe control by optimizing the control forces [21] and Mcnamara et al. used the adjoint method to accelerate it [22]. Recently, very similar to keyframe control, some authors focused on controlling a fine simulation to follow a preview coarser one. Nielsen et al. [5, 6] implemented the guiding control by changing the projection step to solve a minimization equation between the guiding and guided fields. Huang et al. [7] achieved guiding control by applying control forces at sample points to make the velocities follow the guiding velocities. Yuan et al. [8] analyzed the Lagrangian Coherent Structure (LCS) of the guiding velocity field and applied control forces in these LCS regions. Although they can obtain good guiding effects in some cases, generally speaking, their methods cannot guarantee that the guiding always works especially when obstacles can not be represented on the coarse grid for the preview purpose.

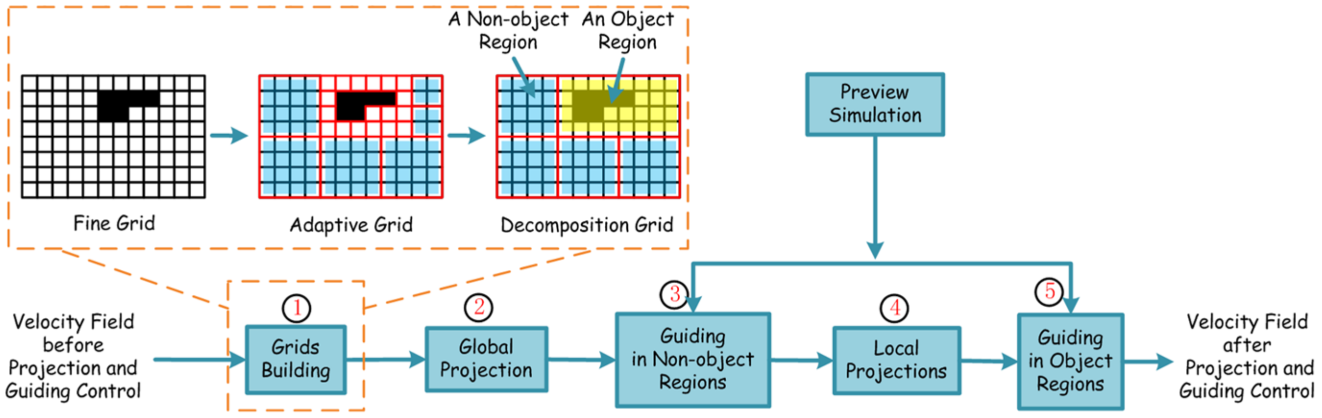

A Non-object Region
An Object Region
Fine Grid
Adaptive Grid
Decomposition Grid
Preview Simulation
Velocity Field before Projection and Guiding Control
1 Grids Building
2 Global Projection
3 Guiding in Non-object Regions
4 Local Projections
5 Guiding in Object Regions
Velocity Field after Projection and Guiding Control

Fig. 1 The algorithmic pipeline of our system. Before fine resolution simulation, a preview one will be quickly conducted and stored. For fine grid simulation, when simulating each frame, we first build the adaptive grid and decomposition grid. To obtain divergence free velocity field, we map the velocities to the adaptive grid and solve the global projection. Then we map the velocities' change back to the fine grid and implement local projection for each local region. Finally, we apply guiding control with the prestored coarse velocity field. To obtain better performance, guiding for nonobject regions goes ahead of the local projection step

Model reduction Model reduction is another method to reduce the projection cost. Some researchers used the model reduction method to handle very high resolution grids [23–25]. However, the use of bases is non-physical and the achieved details are not as good as that of traditional methods. Moreover, when boundary conditions change, all the precomputation must be conducted again. Another approach is to reduce the three-dimensional simulation to a series of two-dimensional ones via simplification [26, 27]. Although they produced visually compelling results in certain instances, in more general cases, the two-dimensional simulations cannot approximate three-dimensional behavior very well.

3 Overview

On the fine grid, the inviscid, incompressible fluids are characterized by the equations

$$\partial \mathbf{u}/\partial t = -\mathbf{u}\nabla \cdot \mathbf{u} - \nabla p + \mathbf{f}, \qquad (1)$$

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

where $\mathbf{u}$ denotes fluid velocity, p denotes pressure, $\mathbf{f}$ is an external force, and the fluid density is assumed to be 1 for the simplicity purpose. This set of equations is usually solved using the operator splitting method [1]. In the splitting scheme, one phase called velocity projection costs the most. In this paper, we change the traditional velocity projection phase to two phases—a global projection phase and a local projection phase – via an adaptive grid. Moreover, we propose a novel guiding control method tightly integrated with our projection method. Our guiding control method is also split to two phases—guiding in nonobject regions and guiding in object regions (nonobject regions and object regions will be described in Sect. 6.1 and an intuitive example is given in Fig. 1).

Based on the splitting scheme, we implement our IAGP-GC method for each frame as follows (also see Fig. 1):

1. Build the adaptive grid and decomposition grid. The adaptive grid is used for velocity projection and the decomposition grid is used for guiding control;
2. Implement global projection on the adaptive grid. The velocity field is mapped from the fine grid to the adaptive grid to implement global projection;
3. Apply guiding control for nonobject regions of the decomposition grid;
4. Implement local projections;
5. Apply guiding control for object regions.

The traditional adding force phase and advection phase are omitted.

4 Projection models

In incompressible fluid simulation, an important step is to make the velocity field divergence-free. This step is called velocity projection. To make a velocity field divergence-free, the following Poisson equation is employed:

$$\triangle p = \nabla \cdot \mathbf{u}^*, \qquad \mathbf{u} = \mathbf{u}^* - \nabla p, \qquad (3)$$

where p is the pressure field, $\mathbf{u}^*$ is the intermediate velocity field, and $\mathbf{u}$ is the result of projection. Poisson equation is usually solved with Dirichlet boundary condition or Neumann boundary condition. To use these two boundary conditions, the simulation domain is usually closed (for example, in a closed box, or a room with the velocity at the open window being fixed).

Alternative models can be derived by minimizing the integral of square of velocity change (also see [5]). We call these models flux-based projection models. In an open domain, which means both boundary velocity and internal velocity can be projected, the flux-based projection model is given by

$$\begin{aligned} &\text{minimize}_{\mathbf{x}} && \int_{\Omega} \left|\mathbf{x}(r)\right|^2 dr, \\ &\text{subject to} && \nabla \cdot (\mathbf{x}(r) + \mathbf{u}^*(r)) = 0, \end{aligned} \tag{4}$$

where Ω is the domain of the entire field including boundaries, x is the change of velocity. This model means that we expect to find a divergence-free velocity field and its Euclidean distance to the original field is minimum. Using flux-based projection models, pressure field is not used, and thus Dirichlet boundary condition or Neumann boundary condition is not necessary.

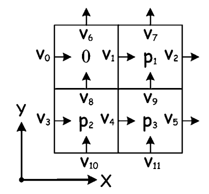
In our integration method, global projection is implemented using flux-based projection models, and local projection and guiding control are based on Poisson equation. In the following sections, Poisson equation and flux-based projection models are discussed in detail.

4.1 Poisson equation

The first property of Poisson equation is that using Poisson equation to do projection, all vorticity except that on the boundaries will be preserved. This is because the divergence part can be represented as the gradient of a scalar field [16, 28], and this gradient field will naturally has no vorticity. We can use this property to decompose the velocity field of a closed domain to its low frequency part (only the global trend and no vorticity) and high frequency part (only the vorticity and no global trend).

The second property of Poisson equation is that projection using Poisson equation is a linear operation. We can rearrange the pressures and velocities in certain order (not unique though) and obtain vectors $\mathbf{p}$ and $\mathbf{v}$, and $\mathbf{v}^*$ is the vector of the intermediate velocity (also see Fig. 2). As the Laplacian matrix is not of full rank, we set the first element of vector $\mathbf{p}$ to zero, and the remaining vector is denoted by $\mathbf{p}'$. A modified corresponding Laplacian matrix, denoted by L_p, can be obtained by expanding Laplacian operation of each cell. Similarly, we can obtain the divergence matrix D_v and the gradient matrix G_p. Then Eq. (3) can be rewritten as

$$\begin{aligned} &L_p \cdot \mathbf{p}' = D_v \cdot \mathbf{v}^*, \qquad \mathbf{p} = \begin{pmatrix} 0 \\ \mathbf{p}' \end{pmatrix}, \\ &\mathbf{v} = \mathbf{v}^* - G_p \cdot \mathbf{p}, \end{aligned} \tag{5}$$



$$\mathbf{p} = \begin{pmatrix} 0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} \quad \mathbf{p}' = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \\ v_{11} \end{pmatrix} \quad G_p = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$L_p = \begin{pmatrix} 1 & 1 & 0 \\ -2 & 0 & 1 \\ 0 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \quad D_v = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

Fig. 2 A 2D grid and its corresponding vectors and matrices

and Eq. (5) can be combined into

$$M = I - G_p \cdot \begin{pmatrix} \mathbf{0} \\ L_p^+ \end{pmatrix} \cdot D_v, \qquad \mathbf{v} = M \cdot \mathbf{v}^*, \tag{6}$$

where L_p^+ is the pseudo-inverse of L_p (which can be computed by a svd-decomposition-based method) and I is an identity matrix. One should note that $\mathbf{0}$ in Eq. (6) is in fact a row vector. The linear property of the Poisson equation can be derived from Eq. (6):

$$M \cdot (\alpha \mathbf{v}_1 + \beta \mathbf{v}_2) = \alpha(M \cdot \mathbf{v}_1) + \beta(M \cdot \mathbf{v}_2), \tag{7}$$

which means blending two velocity fields before projection is equivalent to blending them after projection.

4.2 Flux-based projection model

The discrete flux-based projection model over the adaptive grid in open boundary condition is given:

$$\begin{aligned} &\text{minimize}_{fv} && \textstyle\sum_{i \in faces} W_i \cdot f{v_i}^2, \\ &\text{subject to} && \forall j \in cells, \operatorname{div}(C_j) = 0, \end{aligned} \tag{8}$$

where *faces* is the set of all faces, i is the index of each face, *cells* is the set of all cells, j is the index of each cell, fv is the array of the change of all velocities, W is the array of weight for faces, and $\operatorname{div}(C_j)$ is the divergence of cell C_j. Note that div() is in fact a linear function of fv for each cell. We suggest W_i to be the area of face i (the same as volume-weighted Poisson equation [3]), but one can use other weighting schemes if necessary.

Typically, we assume the adaptive grid to be an octree grid as it is easy and sufficient in most cases. Figure 3 is an example of three-dimensional octree grid. In this example, we assume the area of smaller faces to be 1, thus the area of larger faces is 4. And the constraint for the larger cell using Eq. (8) is $(-4u_1 + u_2 + u_3 + u_4 + u_5 - 4v_1 + 4v_2 - 4w_1 + 4w_2) = 0$.

5 Adaptive grid projection method

In this section, we will describe our adaptive grid projection method via an octree grid. This method is aiming to acceler-

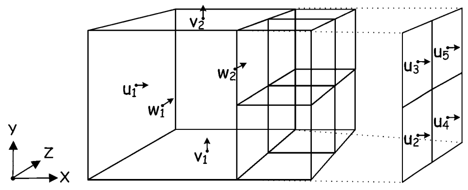
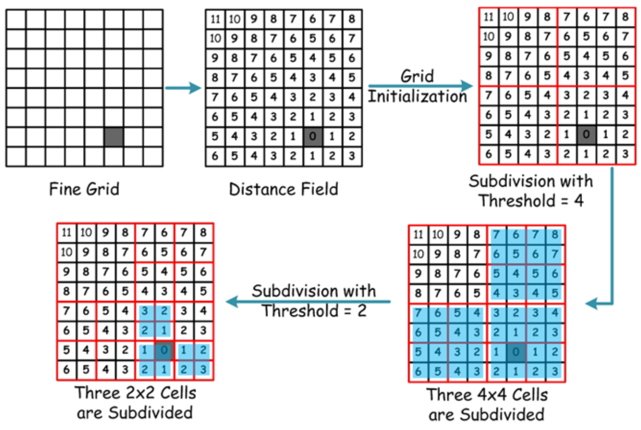


Fig. 3 A node of the octree grid



Fine Grid
Distance Field
Grid Initialization
Subdivision with Threshold = 4
Subdivision with Threshold = 2
Three 2x2 Cells are Subdivided
Three 4x4 Cells are Subdivided

Fig. 4 An example of octree grid generation

ate the most costly step (velocity projection) in a dense grid simulation.

5.1 Octree grid generation

The octree grid is constructed according to a distance field with n thresholds for $n+1$ levels. To compute the distance field, all cells are initialized with a value larger than all the thresholds. Then any cell of the fine grid occupied by obstacles is set to zero for its distance value. Using these zero cells as seeds, the final distance field can be generated via a seed-filling method. After generating the distance field, the octree grid is initialized to be the coarsest regular grid, and then it is recursively subdivided when any leaf node has a fine cell whose distance is smaller than the corresponding threshold. Figure 4 gives an example to generate the octree grid. In this example, we use two thresholds (2 and 4) and three levels of cell size.

5.2 Global projection on the octree grid

To solve the projection on the octree grid, we first compute the velocities according to the velocity field on the fine grid with the following equation:

$$u_c^* = \frac{1}{n} \sum_{f \in faceset} u_f^*, \tag{9}$$

where u_c^* is the velocity of the coarse face, u_f^* is the velocity of the fine face, *faceset* is the set of the fine scale faces, which overlap with the octree face, and n is the number of elements in *faceset*.

Then, using the Lagrange multiplier method, the formulation in Eq. (8) can be rewritten as

$$\text{minimize}_{fv,\lambda} \sum_{i \in faces} W_i \cdot f{v_i}^2 - \sum_{j \in cells} \big(\lambda_j \cdot \text{div}(C_j)\big), \tag{10}$$

where λ is the Lagrange multiplier for each cell of the octree grid. The solution of this equation is the KKT point [29], and one can derive a linear equation $Ax = b$ which can be solved by linear system solvers.

5.3 Local projection

After solving the equation on the octree grid, we will obtain the change fv for all the faces of the octree grid. As we want to preserve the vorticity as much as we can, we simply map the change back to the faces of the fine grid, which overlap with the coarse face and the new velocities are computed by

$$u_f = u_f^* + fv. \tag{11}$$

We do not change velocities of the internal fine faces in each coarse cell as Poisson equation can preserve all internal vorticities [4]. For each local region of coarse cell, we implement projection using the model described in Sect. 4.1. For small-scale cells, each coarse cell's velocities can be rearranged to a column vector and the velocity column vectors of all cells with the same shape can be rearranged to a matrix. Then a GPU-based matrix-matrix multiplication can be employed to solve all the projections very fast. As octree grid has no obstacle in the coarse cells, all the coarse cells have only several shapes (if the largest coarse cell is 8×8, there are at most three shapes, i.e., 2×2, 4×4, 8×8). Alternatively, for larger ones, traditional iterative methods such as PCG can be used instead [29]. This process is also described in Algorithm 1.

6 Our guiding control method

Guiding control is in fact keyframe matching. Users expect the fine resolution smoke to be globally the same shape as the preview one and this matching should be frame by frame. In this section, we will analyze how to control the velocity fields to be matched, and then describe how to implement our novel guiding control method.

Using a low frequency field to control a higher one, one can decompose the latter field into a high frequency part and a low frequency part and apply control only on the low frequency part and then merge the low and high frequency parts [30]. This can be described as follows:

$$F_3 = \alpha L(F_1) + (1-\alpha) L(F_2) + H(F_2), \tag{12}$$

Algorithm 1 Local projection

```
Require: Velocity field before projection $F^*$, obstacles $Obj_f$, irregular grid $Grid$, region sets $RegionSets$.
Ensure: Velocity field after projection $F$.
    // we assume precomputed matrices are $M_1$, $M_2$, $M_3$.
 1: $F \leftarrow 0$; $U_1 \leftarrow [\,]$; $U_2 \leftarrow [\,]$; $U_3 \leftarrow [\,]$;
 2: for each region $R_i \in RegionSets$ do
 3:   if $R_i$ can be computed by GPU matrix-matrix multiplication method then
 4:     $l \leftarrow$ octree level of $R_i$;
 5:     $v_i \leftarrow$ rearrange the velocities of $R_i$;
 6:     $U_l = [U_l \ v_i]$;
 7:   else
 8:     $v \leftarrow$ rearrange the velocities of $R_i$;
 9:     build corresponding matrix for PCGsolver;
10:     $v \leftarrow$ PCGsolver($v$);
11:     rearrange $v$ to the corresponding position of $F$;
12:   end if
13: end for
14: $U_1 \leftarrow M_1 \cdot U_1$; $U_2 \leftarrow M_2 \cdot U_2$; $U_3 \leftarrow M_3 \cdot U_3$;
15: rearrange $U_1$, $U_2$, $U_3$ to the corresponding position of $F$;
16: return $F$;
```

where F_1 is the guiding field, F_2 is the guided field, F_3 is the result, α is the guiding coefficient and $L(F)$ and $H(F)$ are the operations to get the low and high frequency parts of F, respectively.

We decompose the velocity fields using Poisson equation models via an irregular grid which is called "decomposition grid." We stress that the decomposition grid is the key of our guiding control method. According to the linear property described in Sect. 4.1, it is much cheaper to apply guiding control using the following equation:

$$\begin{aligned} F_3 &= \alpha L(F_1) + (1-\alpha)L(F_2) + H(F_2) \\ &= L(F_2) + H(F_2) + \alpha\big(L(F_1) - L(F_2)\big) \\ &= F_2 + L\big(\alpha(F_1 - F_2)\big), \end{aligned} \tag{13}$$

which means that we can first compute the field $\alpha(F_1 - F_2)$ and then add its low frequency part to the guided field. Comparing to Eq. (12), only one decomposition operation is needed in Eq. (13) whereas three are needed in the former case.

In Sect. 6.1, we will describe how to generate the decomposition grid. And in Sect. 6.2, we will describe how to decompose the needed low frequency part field.

6.1 Decomposition grid generation

Before decomposition, we build a decomposition grid to divide the domain to smaller regions. In order to handle the

Algorithm 2 Low frequency part decomposition

```
Require: Coarse velocity field $F_c$, coarse obstacles $Obj_c$, fine velocity field $F_f$, fine obstacles $Obj_f$, decomposition grid $Grid_{dec}$, guiding coefficient $\alpha$, and a flag ($OnlyObjectRegions$ or $AllLocalRegions$).
Ensure: Low frequency part $F_{low}$.
 1: $F_c^* \leftarrow$ Upsample($F_c$);
 2: $F_d \leftarrow \alpha(F_c^* - F_f)$;
 3: for each face $x$ of $F_d$ do
 4:   if ($x$ is not in any region of $Obj_f$) and ($x$ does not overlap with $Grid_{dec}$) then
 5:     $F_d(x) \leftarrow 0$;
 6:   end if
 7: end for
 8: if $flag = OnlyObjectRegions$ then
 9:   $RegionSets \leftarrow$ {all object regions};
10: else if $flag = AllLocalRegions$ then
11:   $RegionSets \leftarrow$ {all local regions};
12: end if
13: $F_{low} \leftarrow$ LocalProjection($F_d$, $Grid_{dec}$, $RegionSets$)
14: return $F_{low}$
```

cases where the obstacles' shapes are different between the low and high resolutions, we need to take special care when building the decomposition grid. The difference of the low and high resolution obstacles should not appear on the local regions' boundaries. And the decomposition grid should not be too dense especially around the obstacles, otherwise the guiding result will be much like upsampling.

An easy way to build such a decomposition grid is to connect all the cells on the octree grid (the adaptive grid used for global velocity projection) smaller than a threshold (such as $4 \times 4 \times 4$ in 3D) to one or several large local regions, which we call "object regions," and preserve each of other cells as a local region, which we call "nonobject region." One can see the zoom-in area in Fig. 1 for an example, the second grid is the adaptive grid for velocity projection and the third grid is the decomposition grid.

6.2 Decomposition of velocity field

Given a coarse velocity field F_c with coarse obstacles Obj_c embedded to guide a fine velocity field F_f with fine obstacles Obj_f embedded and a coefficient α, we can compute the desired low frequency part F_{low} according to the decomposition grid $Grid_{dec}$ as follows (also see Algorithm 1). We first upsample F_c to F_c^*. For faces in the region of any obstacle Obj_{c_i} (region of obstacle includes the obstacle's boundary), the upsampled velocity value should be computed according to the velocity of Obj_{c_i}. For other faces, bilinear interpolation in 2D or trilinear interpolation in 3D can work well. Then the difference field $\alpha(F_c^* - F_f)$ is computed and stored as F_d. Finally, we set all the internal velocities of F_d to zero and implement local projection for each local region of F_d according to the decomposition grid $Grid_{dec}$, and obtain the low frequency part F_{low}.

7 IAGPGC method

In this section, we will describe the framework of our IAGPGC method (also see Algorithm 3).

Before the fine simulation, the preview simulation will be conducted and stored. For fine simulation, the first step is applying boundary condition. In this step, position and velocity of obstacles, position, and density of smoke source and wind forces can be set or changed. The second step is adding force step. Buoyancy force and wind forces can be applied to velocity field. We use the vorticity confinement method [2] to enhance details. Then we use BFECC method [9] to advect the velocity field and employ GPU to accelerate the computation.

Algorithm 3 IAGPGC Method for Each Frame

```
Require: Fine velocity field $F_f$, coarse velocity field $F_c$,
    and scalar field $S$
Ensure: Fine velocity field $F_f$ and scalar field $S$
 1: $Obj_f \leftarrow$ apply boundary condition;
 2: $F_f \leftarrow$ AddForce($F_f$);
 3: $F_f \leftarrow$ AdvectVelocityBFECC($F_f$);
 4: if (first frame) or (any obstacle has changed) then
 5:     $Grid_{oct} \leftarrow$ BuildOctreeGrid($Obj_f$);
 6:     $Grid_{dec} \leftarrow$ BuildDecompositionGrid($Grid_{oct}$);
 7: end if
 8: MapToOctreeGrid($F_f$, $Grid_{oct}$);
 9: GlobalProjection($Grid_{oct}$);
10: MapToFineGrid($Grid_{oct}$, $F_f$);
    // Guiding control in non-object regions: lines 11–16
11: $F_{low} \leftarrow$ LowFrequencyPart($F_c$, $F_f$)
12: for each face $x$ of $F_f$ do
13:     if $x$ overlaps with $Grid_{dec}$ then
14:         $F_f(x) \leftarrow F_f(x) + F_{low}(x)$;
15:     end if
16: end for
    // Local projections: lines 17–18
17: $RegionSets \leftarrow$ {all local regions of $Grid_{oct}$}
18: $F_f \leftarrow$ LocalProjection($F_f$, $Grid_{oct}$, $RegionSets$);
    // Guiding control in object regions: lines 19–23
19: for each face $x$ of $F_f$ do
20:     if $x$ is in any object region of $Grid_{dec}$ then
21:         $F_f(x) \leftarrow F_f(x) + F_{low}(x)$;
22:     end if
23: end for
24: $S \leftarrow$ AdvectScalarBFECC($F_f$, $S$);
```

The following several steps are the main process of our IAGPGC method. According to the obstacles' positions with octree thresholds, we build the adaptive grid, and the decomposition grid is built according to the adaptive grid. If the obstacles have not changed comparing to the last frame, these two grids of the last frame can be reused. The velocity field produced by velocity advection step is mapped from the fine grid to the adaptive grid. We use the PCG method to solve the linear equations derived from Eq. (10). Then velocity field is mapped from the adaptive grid to the fine grid. At this time, we use Algorithm 2 to get the low frequency part F_{low} of the difference field and the flag is set to *OnlyObjectRegions*. As the decomposition method, we use is the Poisson equation, to apply guiding control in nonobject regions, we can simply guide the velocities of the boundary of nonobject regions and do nothing for the internal velocities of nonobject regions. The next is the local projections step. All velocities of the local regions of the adaptive grid are projected using GPU matrix-matrix multiplication method (described in Sect. 5.3). To apply guiding control in object regions, we add the prestored low frequency part F_{low} to the fine velocity field for all object regions.

The final step is advecting scalar field step. We use BFECC method [9] to advect the scalar field (such as temperature and density), and GPU is used to accelerate the computation.

8 Experiment results and comparison

Several results of our adaptive grid projection and guiding control method are discussed in this section.

8.1 Experiment results

Figure 5 shows two examples of our adaptive grid projection method. We observe that our method produces large amount

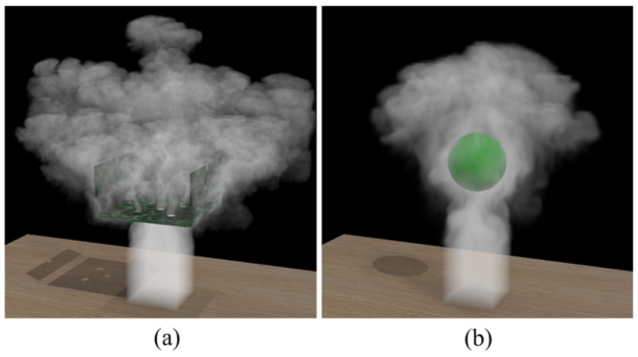


Fig. 5 Two examples of our adaptive grid projection method. (**a**) A 3D example with several holes (resolution 256 × 256 × 256). (**b**) A 3D example with a static sphere (resolution 128 × 128 × 128)

Table 1 Timing information when comparing with Losasso et al.'s method [3], and Lentine et al.'s method [4] deploying a static sphere. All the fine resolutions are $128 \times 128 \times 128$. "Lentine10" means the coarse grid projection method proposed by Lentine et al. [4], "Ours*" means the extension of our flux-based projection model to the coarse grid, "Losasso04" means the volume-weighted Poisson equation described in [3] and "Ours" means our flux-based projection model. The symbol "–" in the second column means we only use three levels of octree cell. "Unkowns/NNZ" is the numbers of unknowns and nonzeros of the derived linear equation $Ax = b$ and "Cost/iterations" is the time cost of the global projection and number of iterations

Method	Coarse or octree	Unkowns/NNZ	Cost/iterations
Lentine10	$32 \times 32 \times 32$	32687/222524	0.28s/150
Ours*	$32 \times 32 \times 32$	133752/493032	0.86s/150
Losasso04	2, 4, –	44799/309489	0.41s/150
Ours	2, 4, –	183294/680172	1.26s/150

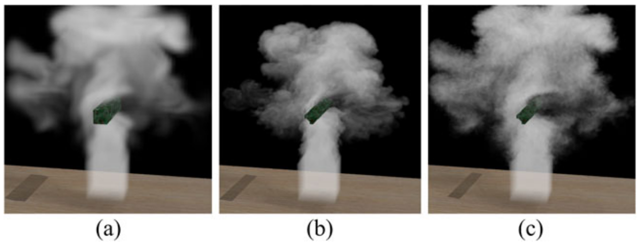


Fig. 6 A 3D example of our guiding control method. (**a**) The preview ($64 \times 64 \times 64$); (**b**) The un-guided fine simulation ($256 \times 256 \times 256$); and (**c**) The fine simulation with guiding coefficient 1.00

of details without any visual artifact. To compare the efficiency of our model with Poisson equation, we deploy a static sphere in the scene (see Fig. 5(b)) and use different models to implement simulations at resolution $128 \times 128 \times 128$. Timing of the projection step is given in Table 1. From the table, we can see that our model has four times as many unknowns and twice as many non-zeros as that in Poisson equation, and our model's cost is about three times as much as that of Poisson equation.

Figure 6 shows an example of our guiding control method. We deploy a stick-shaped moving obstacle and the obstacles between the preview and the fine resolutions are slightly different. We can see the fine un-guided simulation (Fig. 6(b)) is quite different from the preview one (Fig. 6(a)), whereas the guided one (Fig. 6(c)) and the preview are very similar. Our guiding control method also enables users to reuse prestored simulations (Fig. 7(a)) by changing the embedded obstacles (Fig. 7(c)) to certain degree.

8.2 Comparison

To illustrate the advantage of our adaptive grid projection method, we compare our method with coarse projection method proposed by Lentine et al. [4] and the octree

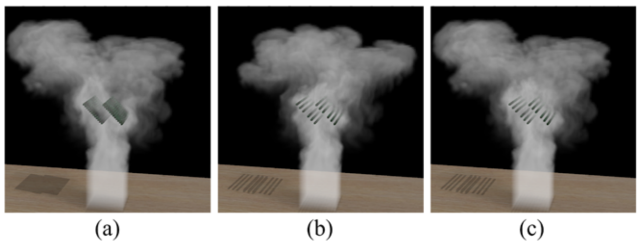


Fig. 7 A 3D example of our guiding control method for reuse (all simulations are $128 \times 128 \times 128$). (**a**) The prestored simulation, (**b**) the simulation without guiding control, and (**c**) the reused simulation with guiding control. Our method enables users to slightly change the obstacles of the scene and produces visually correct results

grid method proposed by Losasso et al. [3]. The projection model of their methods is Poisson equation (or volume-weighted Poisson equation), and we believe this model can only be used in the constrained boundary velocity condition, whereas our projection model can be used in unconstrained boundary velocity condition. The projection step based on our model is a bit slower than that based on Poisson equation, but our method is still a good choice for unconstrained boundary velocity condition. Lentine et al.'s method using a regular coarse grid to do the global projection cannot retain tiny features of obstacles and artifacts will occur in some cases, whereas our method does not have such problems. For example, in Fig. 8(b), the coarse grid is 8×8 coarser than the fine grid, and the velocities in the two coarse cells containing the thin sticks (see the zoom-in area) will be very large and artifacts will occur. In some cases, when using Lentine et al.'s method a coarse cell may be divided into several regions by a tiny obstacle, and each region may not be divergence free after local projection (see Fig. 8(a)). Although they can use octree grid in the local projection step, the local refinement is no better than using dense grid in local region and can never influence the global coarse projection. However, our octree grid is global, and the refinement around the obstacles is done in global projection step.

The other one includes the guiding control methods of Nielsen et al.'s [5, 6], Huang et al.'s [7], and Yuan et al.'s [8]. Nielsen et al. claimed that their method cannot handle varying obstacles very well, whereas using our irregular coarse grid structure we can tackle this problem well. Both Huang et al.'s and Yuan et al.'s methods apply guiding control by adding forces in sample regions. As a result, their methods may not work well when using a coarse projection method (just like ours or Lentine et al.'s [4]) to accelerate the projection, as only sample points on the coarse grid will influence the global projection and the guiding effects will be weakened accordingly.

There are some cases that our guiding control method cannot handle well. When the obstacles are too large and we may need to use verge large object regions, thus the guiding control may not achieve good results.

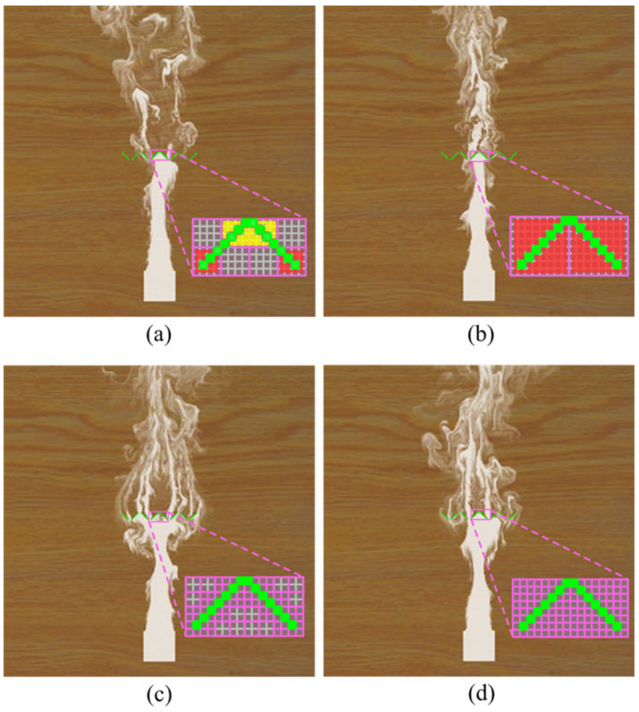


Fig. 8 Comparison with Lentine et al.'s method [4] with tiny objects. (**a**) Lentine et al.'s method with 4×4 coarse cells (**b**) Lentine et al.'s method with 8×8 coarse cells; (**c**) Our octree grid method; (**d**) The fine projection. (**c**) and (**d**) are visually correct, but (**a**) and (**b**) have artifacts. That is because the *yellow areas* in (**a**) may not be divergence free and the *red areas* in (**a**) and (**b**) may have much larger velocity than what the normal velocity should be

9 Conclusion

We have articulated an improved strategy for smoke simulation design over adaptive grids, while simultaneously enabling the functionalities of local projection and guiding control. Our method enables the smoke animation design by offering two additional advantages. First, our new integrated method supports adaptive grid projection that can significantly reduce the computational cost during the velocity projection phase without losing visually interesting details. Second, our integrated solution over adaptive grids can tightly couple guiding control with local projection, which is capable of handling tiny obstacles that are impossible to model with global coarse grids alone during simulation preview. Comprehensive experiments have shown that our integrated approach has the unique advantage of generating turbulent phenomena when interacting with small-scale features of obstacles, and at the same time offering the preview mechanism for efficient large-scale smoke simulation design.

Acknowledgements This research is supported in part by National Natural Science Foundation of China (Nos. 61073078, 61272348, 61202235, 61190120, 61190121, and 61190125), Ph.D. Program Foundation of Ministry of Education of China (No. 20111102110018) and National Science Foundation of USA (Nos. IIS-0949467, IIS-1047715, and IIS-1049448).

References

1. Stam, J.: Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 121–128. ACM Press/Addison-Wesley, New York (1999)
2. Fedkiw, R., Stam, J., Wann Jensen, H.: Visual simulation of smoke. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, pp. 15–22. ACM, New York (2001)
3. Losasso, F., Gibou, F., Fedkiw, R.: Simulating water and smoke with an octree data structure. ACM Trans. Graph. **23**(3), 457–462 (2004)
4. Lentine, M., Zheng, W., Fedkiw, R.: A novel algorithm for incompressible flow using only a coarse grid projection. ACM Trans. Graph. **29**(4), 114 (2010)
5. Nielsen, M.B., Christensen, B.B., Zafar, N.B., Roble, D., Museth, K.: Guiding of smoke animations through variational coupling of simulations at different resolutions. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '09, pp. 217–226. ACM, New York (2009)
6. Nielsen, M.B., Christensen, B.B.: Improved variational guiding of smoke animations. Comput. Graph. Forum **29**(2), 705–712 (2010)
7. Huang, R., Melek, Z., Keyser, J.: Preview-based sampling for controlling gaseous simulations. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '11, pp. 177–186. ACM, New York (2011)
8. Yuan, Z., Chen, F., Zhao, Y.: Pattern-guided smoke animation with lagrangian coherent structure. ACM Trans. Graph. **30**(6), 136 (2011)
9. Kim, B., Liu, Y., Llamas, I., Rossignac, J.: Advections with significantly reduced dissipation and diffusion. IEEE Trans. Vis. Comput. Graph. **13**(1), 135–144 (2007)
10. Selle, A., Fedkiw, R., Kim, B., Liu, Y., Rossignac, J.: An unconditionally stable maccormack method. J. Sci. Comput. **35**(2–3), 350–371 (2008)
11. Molemaker, J., Cohen, J.M., Patel, S., Noh, J.: Low viscosity flow simulations for animation. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '08, Eurographics Association, Aire-la-Ville, Switzerland, pp. 9–18 (2008)
12. Lentine, M., Aanjaneya, M., Fedkiw, R.: Mass and momentum conservation for fluid simulation. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '11, pp. 91–100. ACM, New York (2011)
13. Selle, A., Rasmussen, N., Fedkiw, R.: A vortex particle method for smoke, water and explosions. ACM Trans. Graph. **24**(3), 910–914 (2005)
14. Chen, F., Zhao, Y., Yuan, Z.: Langevin particle: a self-adaptive Lagrangian primitive for flow simulation enhancement. Comput. Graph. Forum **30**(2), 435–444 (2011)
15. Dobashi, Y., Matsuda, Y., Yamamoto, T., Nishita, T.: A fast simulation method using overlapping grids for interactions between smoke and rigid objects. Comput. Graph. Forum **27**(2), 477–486 (2008)
16. Sharif, E., Tong, Y., Kanso, E., Schröder, P., Desbrun, M.: Stable, circulation-preserving, simplicial fluids. ACM Trans. Graph. **26**(1), 4 (2007)
17. Mullen, P., Crane, K., Pavlov, D., Tong, Y., Desbrun, M.: Energy-preserving integrators for fluid animation. ACM Trans. Graph. **28**(3), 38 (2009)

18. Kim, T., Thürey, N., James, D., Gross, M.: Wavelet turbulence for fluid simulation. ACM Trans. Graph. **27**(3), 50 (2008)
19. Schechter, H., Bridson, R.: Evolving sub-grid turbulence for smoke animation. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '08. Eurographics Association, Aire-la-Ville, Switzerland, pp. 1–7 (2008)
20. Narain, R., Sewall, J., Carlson, M., Lin, M.C.: Fast animation of turbulence using energy transport and procedural synthesis. ACM Trans. Graph. **27**(5), 166 (2008)
21. Treuille, A., McNamara, A., Popović, Z., Stam, J.: Keyframe control of smoke simulations. ACM Trans. Graph. **22**(3), 716–723 (2003)
22. McNamara, A., Treuille, A., Popović, Z., Stam, J.: Fluid control using the adjoint method. ACM Trans. Graph. **23**(3), 449–456 (2004)
23. Wicke, M., Stanton, M., Treuille, A.: Modular bases for fluid dynamics. ACM Trans. Graph. **28**(3), 39 (2009)
24. Treuille, A., Lewis, A., Popović, Z.: Model reduction for real-time fluids. ACM Trans. Graph. **25**(3), 826–834 (2006)
25. Pfaff, T., Thuerey, N., Selle, A., Gross, M.: Synthetic turbulence using artificial boundary layers. ACM Trans. Graph. **28**(5), 121 (2009)
26. Rasmussen, N., Quang Nguyen, D., Geiger, W., Fedkiw, R.: Smoke simulation for large scale phenomena. ACM Trans. Graph. **22**(3), 703–707 (2003)
27. Horvath, C., Geiger, W.: Directable, high-resolution simulation of fire on the gpu. ACM Trans. Graph. **28**(3), 41 (2009)
28. Chorin, A.J., Marsden, J.E.: A Mathematical Introduction to Fluid Mechanics. Universitext. Springer, Berlin (1979)
29. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer Series in Operations Research and Financial Engineering. Springer, New York (2006)
30. Thürey, N., Keiser, R., Pauly, M., Rüde, U.: Detail-preserving fluid control. Graph. Models **71**(6), 221–228 (2009)

Qing Zuo received the B.S. degree in computer science and technology from Beihang University, China, in 2010. He is currently working toward the Ph.D. degree in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. His research interests include computer animation, physically based modeling and virtual reality.

Yue Qi received the Ph.D. degree from National University of Defense Technology and then joined Beihang University in 2001. He is a professor in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. His research interests include virtual reality, augmented reality, and computer graphics, with an emphasis on geometry and appearance modeling. He is a member of the IEEE and the ACM.

Hong Qin received the B.S. and M.S. degrees in computer science from Peking University, China. He received the Ph.D. degree in computer science from the University of Toronto (UofT) in 1995. He is a full professor of computer science in the Department of Computer Science at State University of New York at Stony Brook (Stony Brook University). During his years at the University of Toronto, he received UofT Open Doctoral Fellowship. He was also a recipient of the NSF CAREER Award from the US National Science Foundation (NSF), Honda Initiation Award, and Alfred P. Sloan Research Fellow by the Sloan Foundation. Currently, he serves as an associate editor for The Visual Computer, Graphical Models, and Journal of Computer Science and Technology. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer aided geometric design, human-computer interaction, visualization, and scientific computing. Detailed information about him can be found from his website: http://www.cs.sunysb.edu/~qin. He is a senior member of the IEEE and the IEEE Computer Society.