

# Vortex Fluid for Gaseous Phenomena

Sang Il Park<sup>†</sup> and Myoung Jun Kim<sup>‡</sup>

<sup>†</sup> Carnegie Mellon University, Pittsburgh, PA – sipark@cs.cmu.edu

<sup>‡</sup> Ewha Womans University, Korea – mjkim@ewha.ac.kr

---

## Abstract

*In this paper, we present a method for visual simulation of gaseous phenomena based on the vortex method. This method uses a localized vortex flow as a basic building block and combines those blocks to describe a whole flow field. As a result, we achieve computational efficiency by concentrating only on a localized vorticity region while generating dynamic swirling fluid flows. Based on the Lagrangian framework, we resolve various boundary conditions. By exploiting the panel method, we satisfy the no-through boundary condition in a Lagrangian way. A simple and effective way of handling the no-slip boundary condition is also presented. In treating the no-slip boundary condition, we allow a user to control the roughness of the boundary surface, which further improves visual realism.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation

---

## 1. Introduction

In this paper, we present a novel approach to generating gas animation based on the vortex formulation. Our scheme is different from previous methods in computer graphics because we are concerned with a higher level field description. Instead of simulating the individual velocity vector at every fixed grid point, we simulate a set of localized basis velocity fields. By placing and scaling each of the basis fields based on underlying physical laws and user-given boundary conditions, we synthesize a plausible fluid animation. This method allows us to concentrate only on the region of importance that contributes to the whole flow area. Consequently, a small number of computational elements are needed for generating a fluid animation even though the whole flow area is large or unbound.

We choose a purely circulating flow as a basis velocity field, and represent it as a particle, *vortex particle*. This circulating fluid is an appropriate basic building block because the vivid spinning movement and property of incompressibility often make flow look fluid-like; a combination of even a few spinning fields can generate a visually complex swirling movement in the flow. Furthermore, a circulating flow field from each vortex particle automatically satisfies the incompress-

ibility condition, as does the superposition of multiple particles.

Information about the strength of the circulation carried by the particles forms another field over space and time, called a *vorticity field*. Vortex methods deal with the evolution of this vorticity field and consequently make it possible to determine the position and the circulation strength of each vortex particle over time. Although these vortex methods have not yet received much attention in the graphics community, they have been regarded as distinct alternatives to the established computational techniques such as finite difference methods in the computational fluid dynamics community.

Besides the efficiency from the localized computation as mentioned above, the vortex method is suitable for the system based on the Lagrangian description; we track the vortex particles convected along the velocity field freely without grid. This Lagrangian description has several potential advantages over grid-based Eulerian methods. First, it is free of numerical dissipation from which the grid-based methods generally suffer in advection process. Therefore, the generated flow is less damped and does not require any additional reinforcement to keep the fluid moving. Second, the number of numerical elements can be easily adapted to the complex-

ity of the flow. Finally, it is suitable for either open or closed environments.

The Lagrangian description also allows us to represent a boundary condition easily and accurately. Grid-based methods require that boundaries such as walls or obstacles should be discretized along with the grid or the grid should be well aligned with the boundaries in advance. However, our method uses conventional geometry representations such as polygonal objects consisting of line segments in 2D or triangles in 3D for describing the boundary condition.

Our contribution in adapting the vortex method to visual applications is as follows: First, for efficient simulation, we exploit the Lagrangian nature of the vortex method and present an efficient method for evolving the vorticity fields without introducing a system of matrix equations. Second, we resolve the various boundary conditions in a Lagrangian way, in which the boundary geometry is treated accurately. Finally, we address the importance of the no-slip boundary condition for visual appearance of the fluid and present a simple and effective scheme for it. We also suggest a controllable parameter for the roughness of the boundary surface.

The remainder of this paper is organized as follows: We first review related work in Section 2. The basic concept and the formulation of the vortex method are given in Section 3. In Section 4, we present our numerical method in detail. We explain how to satisfy the boundary condition in Section 5. Experimental results are provided in Section 6. In section 7, we discuss limitations and further issues and conclude this paper.

## 2. Related Work

In the computer graphics community, the essence of fluid animation is laid upon visual appearance but not upon strict physical accuracy. Thus, early methods concentrated on developing ad-hocs which can imitate a visual behavior of a fluid based on intuition of the authors [SF92, SF93, Sta97]. However, due to the many control parameters provided, the quality of the resulting animations was heavily dependent upon the skill of an animator. Another problem is the lack of interactivity: it was very hard to make the fluid react on an external force or satisfy the user given boundary conditions.

To overcome those problems, the computer graphics community has adopted physics-based schemes from the Computational Fluid Dynamics. There have been two different points of view in dealing with the fluid dynamics. The first view is to describe the flow field at the fixed computational points, called the Eulerian method. The other method, called the Lagrangian method, concentrates on the motion of an individual particle which roughly imitates a molecule of the fluid. Eulerian methods have been important in simulating fluid dynamics. However, due to the recent development in

*Smoothed Particle Hydrodynamics*(SPH), Lagrangian methods have been gradually attracting more interest.

Based on the Eulerian description, Kajiya and Von Herzen adapted fluid dynamics for the first time for generating fluid animation [KvH84]. Later, Foster and Metaxas presented a realistic gas simulation by considering full 3D dynamics of the fluid [FM97]. Although it could generate realistic movement of the gas, it required a very fine timestep for a stable simulation. By using a semi-Lagrangian formulation in the advection process, Stam guaranteed unconditional stability in simulation, which, thus, enables a large timestep and interactive controls over the fluid [Sta99]. However, this method suffered from the rapidly vanishing movement due to the numerical dissipation. Fedkiw *et al.* alleviated this problem by adding the artificial vorticity confinement term [FSJ01].

Most methods based on the Lagrangian description are based on the SPH. Stam and Fiume were the first in the graphics community to exploit it to simulate fire and gases [SF95]. Desbrun and Cani-Gascuel used it both for modeling a deformable surfaces and solving the governing equation of substances [DCG98, DCG99]. SPH and its extension *Moving Particle Semi-implicit* (MPS) were also used to simulate water [MCG03, PTB\*03].

Differently from SPH, the vortex method has not got much attention since it was first introduced by Gamito *et al.* to the graphics community [GLG95]. Although they showed impressive demonstrations of a turbulent jet stream, their method had several limitations: they limited their simulation to simple 2D cases without any boundary walls or obstacles. Furthermore, their method relied on a fixed grid for the vorticity evolution and did not take advantage of the Lagrangian nature of the vortex method. In this paper, we fully exploit the potential of the vortex method for visual simulation of gasses including full 3D extension, imposition of boundary conditions and control of the boundary surface property. Most recently, Salle *et al.* presented a method, in which they exploited vortex particles for the vorticity confinement on top of the conventional Eulerian method [SRF05]. Our method is different from theirs because our method is entirely based on the Lagrangian description.

## 3. Vortex Method

In this section we will briefly introduce the basic concept and formulations of the vortex method. First, we introduce the definition of the vorticity and its transportation equation describing the fluid dynamics (Section 3.1). Second, the discretization of the vorticity field is presented for numerical computation (Section 3.2). Finally, we deal with calculating the velocity field numerically back from the discretized vorticity field (Section 3.3). For further mathematical details and applications of vortex methods, we refer the readers to the book by Cottet and Koumoutsakos [CK98], and

the survey articles by Leonard [Leo80] and Anderson and Graham [AG91].

### 3.1. Vorticity Equation

Let us start with the definition of vorticity. Given a velocity field  $\mathbf{u}(\mathbf{x}, t) = (u, v, w)$ , an angular velocity  $\Omega$  at a point can be measured by using the *curl operator*  $\nabla \times$  as follows:

$$\Omega(\mathbf{x}, t) = \frac{1}{2} \nabla \times \mathbf{u}(\mathbf{x}, t) = \frac{1}{2} \begin{vmatrix} i & j & k \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{vmatrix}. \quad (1)$$

Conventionally, a vector twice as large as the angular velocity is preferred in order to omit the factor of 1/2. This vector is called as the *vorticity*  $\omega$ :

$$\omega(\mathbf{x}, t) = 2\Omega(\mathbf{x}, t) = \nabla \times \mathbf{u}(\mathbf{x}, t). \quad (2)$$

We turn next to describing the governing equations for fluid dynamics in terms of the vorticity. The fluid flow is governed by two equations:

$$\nabla \cdot \mathbf{u} = 0, \quad (3)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho} \mathbf{f}, \quad (4)$$

where  $\mathbf{f}$  is a force, and  $p$ ,  $\rho$  and  $\nu$  are pressure, density, and viscosity, respectively. The first equation corresponds to the mass conservation, also known as “divergence-free condition” for the velocity field. The second equation, called the Navier-Stokes equation, is for the momentum conservation.

By taking the curl of Equation (4), plugging Equation (3) and rearranging it with some vector algebras, the vorticity transport equation is obtained, which is the governing equation of the vortex method,

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = (\nabla \mathbf{u}) \cdot \omega + \nu \nabla^2 \omega + \frac{1}{\rho} \nabla \times \mathbf{f}. \quad (5)$$

The important point to note here is that the pressure and the associated divergence-free constraint on the velocity disappear. The left hand side of this equation physically means an advection of the vorticity in the Eulerian description. This term can be alternatively represented with the material derivative as  $D\omega/Dt$  (i.e., the rate of change of a quantity in the Lagrangian frame). The right hand side describes the change to the existing vorticity field. The leftmost term among them called as the vorticity stretching term produces local intensification and reorientation of the vorticity. The second term is the diffusion term due to the viscous effect. Finally an external force can change the vorticity field. In short, the vorticity is transported along with the flow while changing the value with respect to the stretching effect, viscosity, and external forces. This concept corresponds well to that of the Lagrangian description.

### 3.2. Vorticity Discretization

We represent the vorticity field as a set of numerical elements in order to facilitate the numerical computation. In grid-based methods, this is done by using regularly distributed nodal points, where the spatial differentiation of a given field is computed using the known spatial relationship between nodes. For vortex methods, we sample the computation domain at discrete points (vortex particles) to facilitate the Lagrangian description of the vorticity field, where each of these vortex particles generates a vorticity field around its center. The discretized vorticity field is approximated as the linear combination of the vorticities of the vortex particles:

$$\omega(\mathbf{x}, t) = \sum_{i=1}^N \zeta(\mathbf{x} - \mathbf{x}_i) \Gamma_i, \quad (6)$$

where  $\mathbf{x}_i$  is the position of the  $i$ -th particle and  $N$  is the number of particles. Each particle has vorticity strength vector  $\Gamma_i$  over the volume (or area in 2D)  $V_i$  associated with the particle: that is,  $\Gamma_i = \int_{V_i} \omega dV \simeq \omega_i V_i$ . The vorticity is distributed around the particle position by the distribution function  $\zeta$  for a smooth interpolation. Here we use the gaussian distribution:

$$\zeta(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} \sigma^D} \exp\left(-\frac{|\mathbf{x}|^2}{2\sigma^2}\right), \quad (7)$$

where  $\sigma$  is a smoothing coefficient and  $D$  is the number of dimension.

### 3.3. Vorticity Inversion Formula

The velocity field  $\mathbf{u}_V$  induced from the vorticity field can be recovered via the solenoidal vector potential field  $\Psi$  known as the *stream function* defined as  $\mathbf{u}_V = \nabla \times \Psi$ . Then, it turns out that

$$\nabla^2 \Psi = -\omega. \quad (8)$$

Once we obtain a stream function from the vorticity field, it is trivial to compute the velocity field. To solve this poisson equation, Gamito *et al.* [GLG95] relied on the grid-based method, which may not fully take the advantages of the vortex method as a Lagrangian method. On the contrary, we solve the poisson equation efficiently in a fully Lagrangian way.

For the solution of the Poisson’s equation, the Green’s function for the Laplacian gives a vorticity-based representation of the stream function, and finally the velocity is determined as

$$\mathbf{u}_V = \mathbf{K} * \omega, \quad (9)$$

where  $\mathbf{K}$  is the convolution kernel commonly known as the Biot-Savart kernel [CK98]. The convolution kernel is subsequently discretized using a quadrature having the positions

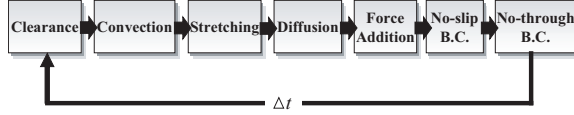


Figure 1: Method Procedure.

of vortex particles as quadrature points. That is,

$$\mathbf{u}_V(\mathbf{x}) = - \sum_{i=1}^N \frac{((\mathbf{x} - \mathbf{x}_i) \times \Gamma_i) q(|\mathbf{x} - \mathbf{x}_i|/\sigma)}{|\mathbf{x} - \mathbf{x}_i|^D}, \quad (10)$$

where  $q(x) = \frac{1}{2\pi}(1 - \exp(-x^2/2))$  for 2D, and  $q(x) = \frac{1}{4\pi}(\operatorname{erf}(x/\sqrt{2}) - \sqrt{2/\pi} x \exp(-x^2/2))$  with the error function  $\operatorname{erf}(s) = \frac{2}{\sqrt{\pi}} \int_0^s (-v^2) dv$  for 3D. Consequently, the velocity field can be obtained as a linear combination of a set of basis velocity fields, each of which is induced by each vortex particle. It is very simple and easy for implementation. It is also interesting to note that each particle generates a divergence-free basis velocity field, and thus, we can always guarantee the divergence-free condition of a resulting velocity field  $\mathbf{u}_V$  even with an arbitrary distribution of the vortex particles. This is one of the strong points of the vortex method from a point of view of controlling the animation.

#### 4. Method Procedure

Now, we deal with the numerical algorithm for evolving the vorticity equation given in Section 3. Provided with an initial state, the vorticity field and its corresponding velocity field are evolved with a timestep  $\Delta t$ . Our method divides the timestep into seven substeps: clearance, convection, stretching, diffusion, force addition, imposition of the no-slip boundary condition and the no-through boundary condition.

The initial velocity state and the boundary geometry are given as input data. As a preprocessing step, we generate initial vortex particles. To uniformly distribute the vortex particles, we first divide the computational domain into a regular grid and then, place a vortex particle at each grid points, where we compute the initial vorticity strengths of the vortex particles using the relationship,  $\omega = \nabla \times \mathbf{u}$ .

For each timestep during the simulation, the first substep is to remove the vortex particles of zero vorticity strength, followed by evolving the vorticity field by convection, stretching and diffusion process without considering boundary effects. Next, we change the vorticity field by adding external forces. From Equation 5, the external forces change the vorticity strengths as much as  $(\frac{1}{\rho} \nabla \times \mathbf{f}) \Delta t$ . If the forces are exerted at the devoid of vorticity, they generate new vortex particles. Sequentially vortex particles are emitted from the boundaries for the no-slip boundary condition. Finally the resulting vorticity field is adjusted to satisfy the no-through

boundary condition. Figure 1 illustrates the procedure of our method.

In the following subsections, we explain the convection, the stretching and the diffusion substeps in detail. We deal with the boundary condition in a separate section (Section 5).

#### 4.1. Convection

The vortex particles are convected with the velocity field. We consider three kinds of velocity fields:  $\mathbf{u}_\infty$  from free stream,  $\mathbf{u}_V$  from vortex particles and  $\mathbf{u}_P$  from the panels of the boundaries which is described in Section 5 later. The free stream means a boundary condition at the infinity. We set it to  $\mathbf{0}$  for a static flow, or a vector representing a uniform flow. The total velocity field  $\mathbf{u}$  is obtained by superposing these three velocity fields:

$$\mathbf{u} = \mathbf{u}_\infty + \mathbf{u}_V + \mathbf{u}_P. \quad (11)$$

We exploit a second-order Adams-Bashforth scheme for time integration as follows:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \left( \frac{3}{2} \mathbf{u}_i^n - \frac{1}{2} \mathbf{u}_i^{n-1} \right), \quad (12)$$

where  $\mathbf{x}_i^n$  is the position of particle  $i$  at time  $n$  and  $\mathbf{u}_i^n$  is the velocity at  $\mathbf{x}_i^n$  at time  $n$ .

Although we are only concerned with the vorticity region for simulation, each timestep of the vortex method requires a heavy computational cost of  $O(N^2)$  with  $N$  vortex particles: for computing the velocity at the position of each particle, we must look up all the other particles according to the velocity inversion formula. From the observation that an influence of a vortex particle to the whole vorticity field drops off rapidly as the distance from the particle increases, we however can accelerate this velocity computation process by grouping the influence of distant particles. We exploit the *Fast Multipole Method* [GR87, Pri94] to realize it. Then, the computational cost for one timestep reduces to  $O(N \log N)$  and it can be further reduced into  $O(N)$  when the group-to-group interaction is considered.

#### 4.2. Stretching

One of the biggest differences between 2D and 3D vortex methods is the existence of the stretching term. In 2D, the stretching term always goes to zero since the direction of the vorticity is perpendicular to that of the velocity. In 3D, the stretching term changes the orientation of the vorticity as well as its value for every timestep:

$$\frac{d\Gamma_i}{dt} = (\nabla \mathbf{u}) \cdot \Gamma_i(t), \quad (13)$$

Since we represent the velocity field as a set of convolution kernel, this stretching term can be calculated by taking the gradient of the kernel analytically.

The vorticity stretching term may result in the instability of the simulation if the vorticity field is not divergence-free; the discretized vorticity field in our method does not guarantee the divergence-free condition. To alleviate this problem, there have been many methods suggested such as relaxation of the divergence of the vorticity field [CK98]. In our method, we rely on the viscous diffusion to maintain a nearly divergence-free vorticity field as discussed in [WL93].

### 4.3. Diffusion

To simulate the diffusion process in a Lagrangian description, we adopt the Random Walk Algorithm [Cho73, GLG95] for the 2D simulation for its efficiency and simplicity in implementation, in which each particle undergoes a Brownian-like motion, and thus the diffusion effect is imitated statistically without increasing the number of vortex particles. We perturb the positions of the vortex particle for every timestep by a random displacement vector whose length has a Gaussian probability distribution with a zero mean and a variance of  $2\nu\Delta t$ .

For the 3D simulation, the Random Walk Algorithm is not suitable since the random perturbation of the particle positions is hard to maintain the divergence-free condition of the vorticity field and consequently, causes the instability during the stretching process. Instead, we apply the method of Particle Strength Exchange (PSE), which approximates the Laplacian operator  $\nabla^2$  by distributing one's strength to its neighbors [CK98]. This method is known to reduce the total kinetic energy gradually, and make the simulation robust while conserving the circulation [WL93]. Then, the diffusion process is simulated as follows:

$$\frac{d\Gamma_i}{dt} = \nu \nabla^2 \omega(\mathbf{x}) = \frac{2\nu}{\sigma} \sum_{j=1}^{N_p} (V_j \Gamma_j - V_i \Gamma_i) \zeta(\mathbf{x}_i - \mathbf{x}_j), \quad (14)$$

where  $V_i$  is the volume associated with particle  $i$ . In practice, this diffusion process can be localized by ignoring the interaction between two particles when their distance is bigger than a threshold. In our implementation, we use  $5\sigma$  for the threshold.

The vortex method has little numerical dissipation during the evolution. As a result, the generated fluid tends to keep moving too long, which is desirable for generating dynamic scenes. However, that can be undesirable for controlling the fluid motion. We provide an artificial damping effect to the vorticity field by adjusting the vorticity strengths of the vortex particle: we gradually reduce the vorticity strengths over time. This process visually makes the fluid look more viscous and also helps to maintain the total number of particles used in one scene to be uniform.

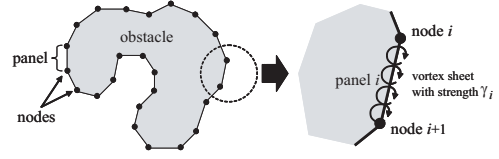


Figure 2: Definition of nodes and panels.

## 5. Boundary Condition

In this section, we explain how to handle the effect of walls and obstacles in a Lagrangian frame. For the simplicity of explanation, we first deal with the no-through condition in Section 5.1 for 2D and its extension to 3D in Section 5.2, and then, the no-slip condition is considered in Section 5.3.

### 5.1. No-through Boundary Condition in 2D

For enforcing no-through boundary condition, we adapt the concept of the *panel method* from the classical airfoil theory, which was originally invented by Hess and Smith [HS64]. The basic idea of this method is to distribute sources or vortices on the surface of the geometry in order to cancel the normal component of the velocity to the surface for the no-through boundary condition. In our method, we only use vortices to satisfy the boundary condition because it facilitates satisfying the no-slip boundary condition easily, as described later in Section 5.3.

The panel method is grid-free and able to represent the boundary walls and obstacles more rigorously: it facilitates to describe the boundaries as a set of line segments. It, thus, represents the geometry as accurately as possible. Additionally, for the geometry with a constant shape, our method enables us to pre-compute all the necessary coefficients related to the boundary geometry in advance. In runtime process, it eventually costs just as much as a matrix-vector multiplication. Moreover, it does not require any extra cost for the rigid transformation of the geometry such as translation or rotation.

We turn now to the details of our implementation. Given the curves representing the boundary of walls or obstacles, we decompose them into a set of line segments, the *panels*. Each panel is modelled as an infinite row of vortices of an equal strength as illustrated in Figure 2. This concept is called the *vortex sheet* which is commonly used in airfoil theory for generating a singularity in the velocity field [Whi94]. The vortex strength is allowed to vary from panel to panel so that each panel has its own panel strength  $\gamma$ , which is defined as the circulation per unit length of the sheet:  $\gamma = d\Gamma/dx$ . By determining the strength of the every panel properly, we can enforce the no-through boundary condition.

Let us first consider the influence of one single panel to the velocity field before dealing with a set of panels. A potential

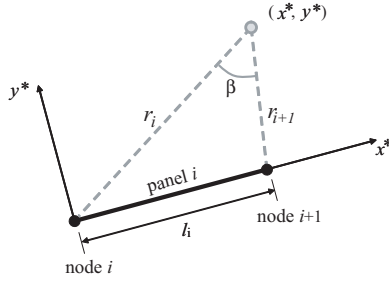


Figure 3: Local panel coordinate system.

field  $\phi$  generated from a panel of strength  $\gamma$  is obtained as follows:

$$\phi(\mathbf{x}) = - \int_{\text{panel}} \frac{\gamma}{2\pi} \theta(s) ds, \quad (15)$$

where  $s$  is a length parameter along the panel and  $\theta$  is an angular component of a position  $\mathbf{x}$  in a spherical coordinate system centered at  $s$ . The velocity can be computed from the potential by using the relationship between them:  $\mathbf{u}(\mathbf{x}) = (u, v) = \nabla\phi = (\partial\phi/\partial x, \partial\phi/\partial y)$ . Without loss of generality, we define a local coordinate system as illustrated in Figure 3 such that the axis  $x^*$  is coincident with the tangential direction of the panel and so the axis  $y^*$  is with the normal direction for simplicity. From Equation (15), the corresponding local velocity field  $\mathbf{u}^*(x^*, y^*) = (u^*, v^*)$  generated from the panel becomes

$$u^* = - \frac{\gamma}{2\pi} \int_0^l \frac{y^*}{(x^* - s)^2 + y^{*2}} ds = \frac{\gamma\beta}{2\pi}, \quad \text{and} \quad (16)$$

$$v^* = - \frac{\gamma}{2\pi} \int_0^l \frac{x^* - s}{(x^* - s)^2 + y^{*2}} ds = \frac{\gamma}{2\pi} \ln \frac{r_{i+1}}{r_i}, \quad (17)$$

where  $l$  is the panel length,  $\beta$  is the angle subtended by the panel at the point  $(x^*, y^*)$ , and  $r_i$  is the distance from node  $i$ . Note that  $\beta$  becomes singular when the measuring point approaches to the panel. That is, from the outside,  $\beta$  goes to  $\pi$ . On the contrary,  $\beta$  becomes  $-\pi$  from the inside. In our case, we focus on only the outside of the obstacles, we always take  $\beta = \pi$  on the panel. Another singularity making the  $v^*$  component infinite happens when approaching to the edge of the panel. We alleviate this problem by adding a small regulating factor  $\varepsilon$  to both of  $r_i$  and  $r_{i+1}$ . Then, the altered component  $v^{**}$  becomes

$$v^{**} = \frac{\gamma}{2\pi} \ln \frac{r_{i+1} + \varepsilon}{r_i + \varepsilon}. \quad (18)$$

Since the resulting velocity is proportional to the strength of vortex sheet, we represent  $\mathbf{u}^*$  as  $\mathbf{u}^* = \tilde{\mathbf{u}}^* \gamma$ , where  $\tilde{\mathbf{u}}^*$  is the velocity field induced by the panel of a unit strength.

For generating the velocity field from a set of panels, we

superpose the velocity field of each panel. That is,

$$\mathbf{u}_P(\mathbf{x}) = \sum_{i=1}^{N_P} \mathbf{T}_i \tilde{\mathbf{u}}_i^* (\mathbf{T}_i^{-1} \mathbf{x}) \gamma_i, \quad (19)$$

where  $\mathbf{u}_P$  is the resulting velocity field from the set of panels,  $N_P$  is the number of panels,  $\mathbf{T}_i$  is the transformation matrix from the world coordinate to the local coordinate associated with  $i$ -th panel,  $\tilde{\mathbf{u}}_i^*$  is the local velocity field by the  $i$ -th panel of a unit strength, and  $\gamma_i$  is the  $i$ -th panel's strength of vortex sheet.

Now, we deal with the no-through boundary condition by determining the strength of each panel such that the normal components of the existing velocity field to the panels are cancelled to be zero. We, however, can not impose the boundary condition at all the points on the panel exactly since the velocity may vary from point to point continuously while we use discrete panel strengths. Instead, we choose the points located at the midpoint of each panel, and impose the no-through boundary condition at those points at least. Thus, the size of the panel should be determined according to required resolution. If a more strict boundary condition is required, smaller panels should be used. In practical, it is enough to have a resolution of the panel be similar to the particle distribution in our experiments.

Let  $\tilde{\mathbf{u}}_{ij}$  be a velocity at the midpoint of the  $j$ -th panel induced by the  $i$ -th panel of a unit strength of the vortex sheet, and  $\mathbf{u}^{in}$  be the existing velocity field before imposing the boundary condition, that is,  $\mathbf{u}^{in} = \mathbf{u}_\infty + \mathbf{u}_V$ . Then, to cancel the normal component of the velocity  $\mathbf{u}_j^{in}$  at the mid point of the  $j$ -th panel,

$$\mathbf{u}_j^{in} \cdot \mathbf{n}_j = - \left( \sum_{i=1}^{N_P} \tilde{\mathbf{u}}_{ij} \gamma_i \right) \cdot \mathbf{n}_j, \quad (20)$$

where  $\mathbf{n}_j$  is the normal vector of the  $j$ -th panel. Equation (20) leads to a linear system for calculating the panel strengths:

$$\mathbf{A} \boldsymbol{\gamma} = \mathbf{b}, \quad (21)$$

where  $A_{ji} = - \left( \sum_{i=1}^{N_P} \tilde{\mathbf{u}}_{ij} \right) \cdot \mathbf{n}_j$  means an influence of the  $i$ -th panel to the  $j$ -th panel and  $\mathbf{b}$  is a vector of the normal component of the input velocity, that is  $b_j = \mathbf{u}_j^{in} \cdot \mathbf{n}_j$ .

The vector  $\mathbf{b}$  has to be updated for every time step as input velocity varies dynamically over time. However, as long as the geometry of the boundary remains unchanged, the matrix  $\mathbf{A}$  stays constant. Moreover, it is not affected by the rigid transformation of the geometry such as translation and rotation. These properties allow us to pre-compute the inverse of the matrix in advance and exploit it to evaluating the strengths  $\boldsymbol{\gamma}$  during runtime.

This method, however, does not guarantee the conservation of the circulation along the boundary, known as the Kelvin's theorem [Cur93]. In fact, in some cases, this may result in a

rapid spinning movement around the boundary while canceling the in-flow velocities. We, thus, impose the zero circulation criterion around the boundary:  $\sum_{i=1}^{N_p} \gamma_i = 0$ . This consequently adds one more equation to the system: Finally we have the  $N_p$  unknowns with the  $N_p + 1$  equations. It is an over constrained system, which can be easily solved in various ways. In our implementation, we exploit the pseudo-inverse of  $\mathbf{A}$  for finding a least square fitting of the strength values.

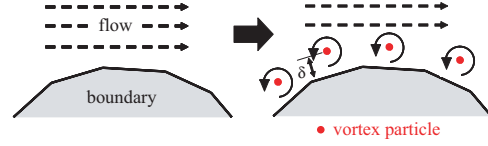
### 5.2. No-through Boundary Condition in 3D

The panel method is extended for 3D applications. Instead of line panels, we use triangular panels to represent the boundary surface. The overall procedure is similar with that of 2D case; a panel is covered with an infinite arrays of vortices of an equal strength, and the strength of each panel is determined for every timestep such that the normal component of the existing velocity field to the panel is cancelled. The velocity field induced by a panel is computed by using the Gaussian quadrature. According to the definition of the vortex sheet, the panel strength vector should be parallel to the tangential plane of the panel, which reduces the number of unknown components of a strength vector of each panel from three to two. Unfortunately, we have only one condition of cancelling the normal component for each panel, which is not enough to determine those two components of the strength vector.

On top of the normal component condition, we impose an additional condition of cancelling the tangential components of the velocity field to the panel surface. Cancelling the tangential components is well-known to be equivalent with cancelling the normal component as long as the vorticity field from the vortex particles satisfies the divergence-free condition [CK98]. In fact, in an ideal case with numerous well-distributed vortex particles, cancelling only the tangential components is enough to satisfy the no-through boundary condition [PWS\*02]. However, we cancel the normal components and the tangential components of the velocity field to the panels simultaneously to make the simulation robust even in the case of using a small number of irregularly-distributed particles as well. Consequently, we have  $2 \times N_p$  unknowns with  $3 \times N_p$  equations:  $N_p$  equations for cancelling the normal component and  $2 \times N_p$  for the tangential components. The pseudo-inverse is exploited to find the solution.

### 5.3. No-slip Boundary Condition

For viscous fluids, the flow velocity in the proximity of the surface should be equal to that of the surface, which creates visually interesting flow patterns around the boundaries such as backflows (separations), and wakes. Because of the complexity of these phenomena, this no-slip boundary con-



**Figure 4:** Imposition of no-slip boundary condition by emitting vortex particles

dition has not been well treated in the graphics community although it is clearly helpful to increase the visual realism.

It is well known in Fluid Dynamics that the vorticity generated on the surface causes these phenomena [Whi94]. In the vortex method, the no-slip condition can be realized by emitting the vorticities of panels to the outside; the emitted vorticities cancel the velocity of the flow tangent to the surface (slip velocity) [PW00, PWS\*02].

In our method, we choose a simple approach: we emit the vorticity from the panel by placing new vortex particles a small distance of  $\sigma$  off the body for every several timesteps as shown in Figure 4. The number of inserted particles is determined to have the similar resolution to that of the existing vortex particles. In our implementation, we roughly generate one particle per panel. The vorticity of each inserted particle is determined with respect to the strength and the area (or length in 2D) of its nearest panel as follows:

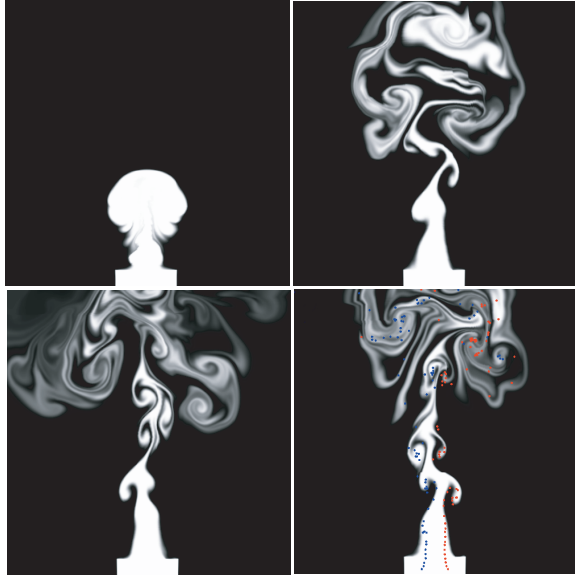
$$\omega = (\tau \cdot \gamma \cdot a) \Delta t, \quad (22)$$

where  $\tau$ ,  $0 < \tau < 1$  is a coefficient to determine the rate of the viscosity emission and  $a$  is area of the nearest panel. Note that the strength of the panel at the previous timestep is used here because that of current timestep is not available yet when the no-slip boundary condition is considered.

Although  $\tau$  is physically related to many factors such as viscosity  $\nu$ , offset  $\delta$ , and the surface roughness, we provide it as a control parameter, and allow an animator to adjust the degree of the boundary effects (the thickness of the boundary layer). Visually this effect is noticeable: ranging from a “slick wall” to a “rough wall”; it generates different flow patterns.

## 6. Results

We have generated a number of animations using our method. The experiments were performed on the PC with PentiumIV 3.0GHz CPU and 1GB Main memory. We rendered the simulation results in two different ways. For the comparison purpose with the previous methods, we developed a grid-based rendering system, in which a regular grid is used for storing and advecting the density of the smoke in a Eulerian manner. In this case, we first evaluate the velocities at every node of the grid by the vortex method, and then, advect the density in a semi-Lagrangian way [Sta99]



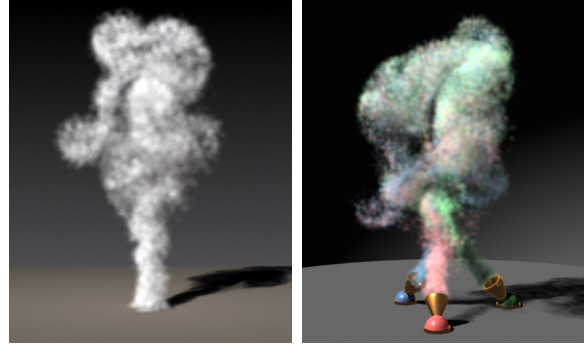
**Figure 5:** Rising Smoke in 2D generated by our method. Even with the small number of computational elements (200 vortex particles in this case), it can express a complex, swirling movement of the gaseous phenomena.



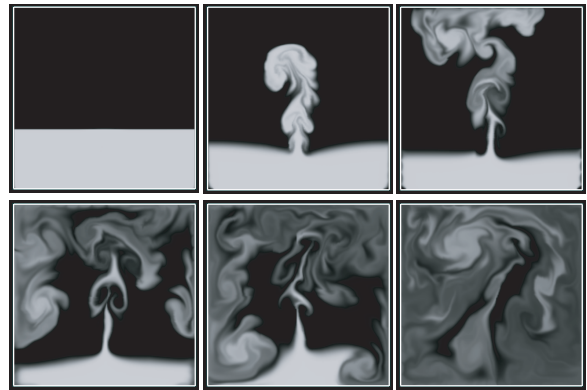
**Figure 6:** Cloud-like smoke in 2D generated by using the particle rendering system for real-time application

from node to node. On the other hand, in some experiments, we adapted a particle rendering system to visualize smoke while exploiting the Lagrangian nature of the vortex method, where the velocity of each particle is computed directly from the vortex particles.

We first demonstrate how an extremely small number of vortex particles can generate a complex scene. We simulate a rising smoke in 2D as shown in Figure 5 with the  $500 \times 500$  rendering grid. The number of vortex particle used in this scene is 200 as shown in the lower right image, where each dot in the figure represents a vortex particle. The color of dots indicates the vorticity strength: from purple to red, the vorticity of rotating counter-clockwise becomes stronger. Blue denotes the opposite direction. It also shows that it does not suffer from the numerical dissipation so that the generated scene looks more dynamic.



**Figure 7:** Rising Smoke in 3D.



**Figure 8:** An example of the simulation in a bounded domain.

For the second experiments, we implement a particle rendering system for real-time applications such as games or live broadcasts. For the rendering purpose, we insert additional particles with transparent texture which are only advected with the velocity field without affecting it: we call it as a *rendering particle* to prevent any confusion with the vortex particles. Figure 6 shows an example of the 2D result of our particle rendering system, where 200 vortex particles and 900 rendering particles are used. In this case, we imposed the artificial damping term as explained in Section 4.3 to gather and slow down the smoke at the upper side.

Figure 7 shows 3D examples. We used the software renderer of MAYA<sup>TM</sup> for rendering particles. The right animation is generated by using 2400 vortex particles and 30000 rendering particles. The left one demonstrates an example of a complex 3D scene, where 6000 vortex particles and 80000 rendering particles are used.

Now, we demonstrate the effectiveness of our method of considering various boundary conditions. First, we demonstrate the imposition of the no-through boundary condition. As shown in Figure 8, we have performed gas simulation in



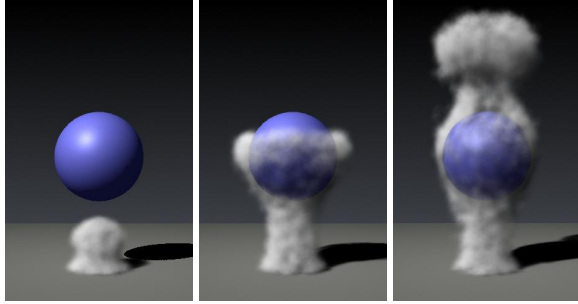


Figure 9: Rising smoke swirling around a sphere.

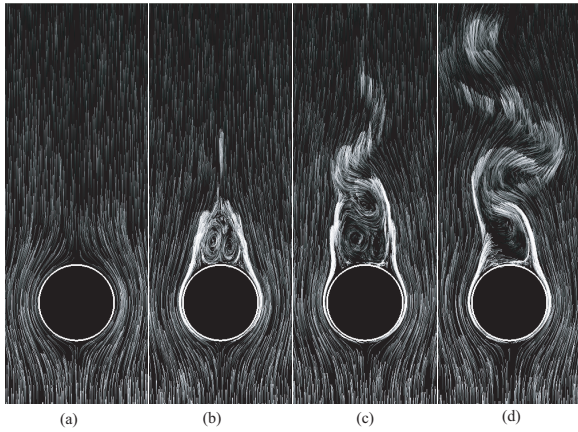


Figure 10: Imposing boundary conditions: (a) no-through boundary condition; no-slip boundary condition at time (a)  $t=1$ ; (c)  $t=2$  and (d)  $t=3$ .

a closed environment: we filled gas in a box and imposed upward force at the middle of the gas-filled region. The box is modelled as a set of panels, where each of 4 sides of the box is discretized into 30 panels. In Figure 9 shows a rising smoke swirling around a sphere in 3D. The sphere is discretized with 288 triangular panels.

Satisfaction of the no-slip boundary condition produces visually interesting flows, which is illustrated in Figure 10: only with the no-through boundary condition, the flow passes around the object smoothly (Figure 10(a)). However, as no-slip boundary condition is started to be taken into account, the backflows first occur (Figure 10(b)) and then, sequentially wakes happen at the back side of the objects (Figure 10(b),(c)). Figure 11 shows the difference between two boundary conditions clearly.

We provide  $\tau$  for controlling the roughness of the boundary surface. The effects of controlling  $\tau$  were experimented as shown in Figure 12. Notice that the range of influence of the no-slip condition is increasing along with  $\tau$ . With a small  $\tau$ , the wake and turbulence at the back side of the object are

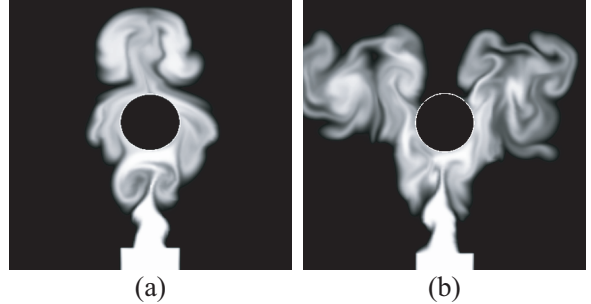


Figure 11: Difference between two boundary conditions: (a) no-through boundary condition and (b) no-slip boundary condition.

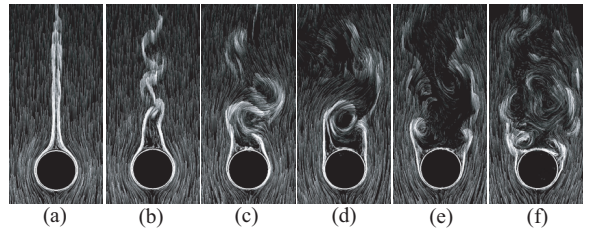


Figure 12: Changes in visual appearance with respect to  $\tau$ : (a)  $\tau = 0.05$ ; (b)  $\tau = 0.15$ ; (c)  $\tau = 0.3$ ; (d)  $\tau = 0.5$ ; (e)  $\tau = 0.75$  and (f)  $\tau = 1.0$ .

little developed as if the boundary surface were smooth. On the other hand, a large  $\tau$  creates strong swirling and turbulent flows, that can be observed around the rough surfaces in the real world.

Table 1 gives the overall performance of our method excluding rendering time. As expected, the computational cost is almost proportional to the number of vortex particle due to the fast multipole method. The computational cost of the 3D simulation is higher than that of the 2D simulation since the vortex particle carries a 3D vector for the vorticity strength rather than a scalar. Moreover, the additional vorticity stretching process is required in 3D.

Table 1: Computation times [sec./frame] for evolving vorticity field obtained by varying the number of vortex particles.

# of particles	100	500	1000	5000	10000	20000
2D	0.0094	0.0134	0.0186	0.0856	0.2072	0.5062
3D	0.0099	0.0396	0.0736	0.9964	2.7552	7.2842

## 7. Discussion and Conclusion

In this paper, we present a new method for generating animation of gaseous phenomena based on vortex method. The vortex method is a powerful and sophisticated numerical method for simulating fluid flow such as gas and smoke. By fully exploiting the advantages of the vortex method, we have customized it for the visual simulation of the fluid. As a result, the generated flow is visually more dynamic and conserves its rotating movement better. Due to the panel method, we have satisfied different boundary conditions. Especially, we have obtained visually interesting flows such as backflows and wakes by properly treating the no-slip boundary condition. It also provides a visual illusion of controlling the roughness of the boundary surface. Our method is efficient since only the vorticity carrying regions are taken into account during the simulation.

There already exist grid-free Lagrangian methods for flow simulation such as SPH in computer graphics. However, it has been troublesome to them to satisfy the divergence-free condition since it is closely related to the distribution of the computational elements. In our method, we always satisfy this condition regardless of the distribution of the vortex particles. Additionally, although SPH have given adaptivity in simulation to some extent, they basically require the computational elements to cover the whole computational domain.

The panel method is a promising technique for adjusting the existing flow field while satisfying the divergence-free condition of the flow. However, in our implementation, it is only accurate at the mid point of each panel and gets inaccurate approaching the edge of panels. Although practically we have not suffered too much from that inaccuracy, it can be improved by applying a continuously varying panel strength along the boundary rather than discrete ones we have used. However, it would be computationally expensive.

Although we are concerned with a  $N$ -body problem, the computational cost is not  $O(N^2)$ , but  $O(N \log N)$  due to the fast multipole method, in which influences from distant particles are treated together as a group. If necessary, it can be further accelerated to  $O(N)$  when group-to-group interactions are exploited. The panel method is also accelerated with the similar manner into  $O(N \log N)$  or  $O(N)$ . For rendering, we have to evaluate the velocity for every rendering particles or every nodal points of the Eulerian grid for rendering. In this case, the rendering cost is  $O(M \log N)$  or  $O(M)$  with  $M$  rendering particles or nodal points. Since  $N$  is generally less than  $M$ , it is competitive or even faster than the conventional physically based methods.

## Acknowledgments

We would like to thank Jessica Hodgins and Doug James for their support and comments. Partial support for the first author was provided by NSF CNS-0203912 and Information

and Telecommunication National Scholarship Program of the IITA (Institute of Information Technology Assessment).

## References

- [AG91] ANDERSON C. R., GREENGARD C.: Vortex dynamics and vortex methods. In *Lectures in Applied Mathematics* (1991), vol. 28, American Mathematical Society, Providence.
- [Cho73] CHORIN A. J.: Numerical study of slightly viscous flow. *Journal of Fluid Mechanics* 57, 4 (1973), 785–796.
- [CK98] COTTET G.-H., KOUMOUTSAKOS P. D.: *Vortex Methods: Theory and Practice*. Cambridge University Press, 1998.
- [Cur93] CURRIE I.: *Fundamental Mechanics of Fluids*. McGraw Hill, 1993.
- [DCG98] DESBRUN M., CANI-GASCUEL M.-P.: Active implicit surface for animation. In *Proceedings of Graphics Interface* (1998).
- [DCG99] DESBRUN M., CANI-GASCUEL M.-P.: *Space-time adaptive simulation of highly deformable substances*. Tech. rep., INRIA, 1999.
- [FM97] FOSTER N., METAXAS D.: Modeling the motion of a hot turbulent gas. In *Proceedings of SIGGRAPH 1997* (1997), pp. 181–188.
- [FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001* (2001), Fiume E., (Ed.), pp. 15–22.
- [GLG95] GAMITO M. N., LOPES P. F., GOMES M. R.: Two-dimensional simulation of gaseous phenomena using vortex particles. In *the 6th Eurographics Workshop on Computer Animation and Simulation* (1995), vol. 28, Springer-Verlag, pp. 3–15.
- [GR87] GREENGARD L., ROKHLIN V.: A fast algorithm for particle simulation. *Journal of Computational Physics* 73 (1987), 325–348.
- [HS64] HESS J. L., SMITH A. M.: Calculation of non-lifting potential flow about arbitrary three-dimensional bodies. *Journal of Ship Research* 8 (1964), 22–44.
- [KvH84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume density. In *Proceedings of SIGGRAPH 1984* (1984), pp. 165–174.
- [Leo80] LEONARD A.: Review: Vortex methods for flow simulation. *Journal of Computational Physics* 37 (1980), 289–335.
- [MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation* (2003), pp. 154–159.
- [Pri94] PRINGLE G. J.: *Numerical Study of Three-Dimensional Flow using Fast Parallel Particle Algorithm*. PhD thesis, Department of Mathematics, Napier University, Edinburgh, 1994.
- [PTB\*03] PREMOŽE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. In *Proceedings of EUROGRAPHICS 2003* (2003), vol. 22.
- [PW00] PLOUNHANS P., WINCKELMANS G. S.: Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal of Computational Physics* 165 (2000), 354–406.
- [PWS\*02] PLOUNHANS P., WINCKELMANS G. S., SALMON J. K., LEONARD A., WARREN M. S.: Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at  $re=300, 500$  and  $1000$ . *Journal of Computational Physics* 178 (2002), 427–463.
- [SF92] SHINYA M., FOURNIER A.: Stochastic motion - motion under the influence of wind. In *Eurographics '92* (1992), pp. 119–128.
- [SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Proceedings of SIGGRAPH 1993* (1993), pp. 369–376.
- [SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of SIGGRAPH 1995* (1995), pp. 129–136.
- [SRF05] SALLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (SIGGRAPH 2005)* 24, 3 (2005).
- [Sta97] STAM J.: *A general animation framework for gaseous phenomena*. Tech. Rep. R047, ERCIM Research Report, 1997.
- [Sta99] STAM J.: Stable fluid. In *Proceedings of SIGGRAPH 1999* (1999), pp. 121–128.
- [Whi94] WHITE F. M.: *Fluid Mechanics*. McGraw Hill, 1994.
- [WL93] WINCKELMANS G. S., LEONARD A.: Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *Journal of Computational Physics* 109 (1993), 243–273.