Swirling Images

Yootai Kim*
The Ohio State University

Raghu Machiraju†
The Ohio State University

Abstract

In this paper, we present a method that uses a point vortex method to produce swirling-like effects in an image plane. Two-dimensional incompressible inviscid Navier-Stokes equations are formulated in terms of vorticity, which measures local rotation of a fluid. Because fluid vorticity is localized to specific regions of the flow, the vorticity formulation reduces the mathematical description to its essential components. Thus, vorticity formulation provides an intuitive control for swirling effects. Then, the Vortex-In-Cell(VIC) scheme is used to solve the equations numerically. In methods based on the VIC schemes, the vorticity field is approximated by vortex particles (or vorticles in our terminology), and a computational grid is used for the fast computation of the velocity field populated with vorticles. This particle-based hybrid approach provides inherent adaptivity and flexible control for computer graphics applications. We demonstrate the efficacy of our method to produce realistic swirling animations from images.

1. Introduction

Swirling motion as typified by a vortex, is the most distinct among fluid motion. In this paper, we introduce an effective method to produce and control swirling effects. Fluid effects are often used in the electronic game and film industry. The fluid effects currently sought by the game and film industry are increasingly more sophisticated and their realization in software has been quite demanding. Depicting various fluid effects thus has been a very challenging problem for computer graphics. In practice, particle-based methods and procedural noise-based methods have been the major techniques employed to generate flow effects largely due to their relative ease-of-control and inherent interactivity [6, 11, 14, 16, 17, 20, 22]. However, particle-based methods are limited in the kind of realism that can be achieved. They require a large amount of effort by skilled animators to generate the desired effects.

For the past few years, many techniques from the Computational Fluid Dynamics (CFD) community have been applied towards the visual simulation of fluid to generate realistic computer generated fluid animations [7–9, 15, 23]. Many of these techniques rely on the Navier-Stokes equations. These formulations define the physical laws which govern the motion of an incompressible Newtonian fluid. However, only the traditional velocity formulation has been used so far by researchers in computer graphics. Often ignored is the vorticity formulation of the Navier-Stokes equations which we employ in this paper. We use vortex methods which are based on the Lagrangian description of the vorticity transportation equation and the approximation of the vorticity field using vortex particles with compact support.

With vortex methods one can create images shown in Figure 4, 5, and 6. It is relatively easy to use vortex methods to achieve the effects depicted in those images. The essential aspect of vortex methods is the focus on point-wise vorticity of a flow field. One reason why vortices are important in fluid motion is that they are often the sources for an incompressible flow field in the domain. Therefore, the time evolution of vorticities dictates the essential physics of the unfolding flow. The situation is analogous to the gravitational field induced by the planets. Mass is often concentrated in comparatively few places, and it is easier to focus on the evolution of the planets rather than the evolving gravitational field. As a result, computational resources are expended most in regions of greatest physical interest by focusing on the evolution of vortices. [10] In vortex methods, an incompressible flow field can be thought of in terms of the mutual interaction of vortex elements (action-at-a-distance model), rather than in terms of the velocity field itself (field model) [3]. Thus, a continuous vortex field is initially represented by vortex particles or vorticles and then the vortex field is iteratively computed to evolve the particles. Nonlinear advection terms replaced with a set of ordinary differential equations for the trajectories of the Lagrangian elements, result in robust schemes with minimal numerical dissipation. However, one drawback of vortex methods is the heavy computational burden towards the de-

* e-mail: yootai@cis.ohio-state.edu
† e-mail: raghu@cis.ohio-state.edu

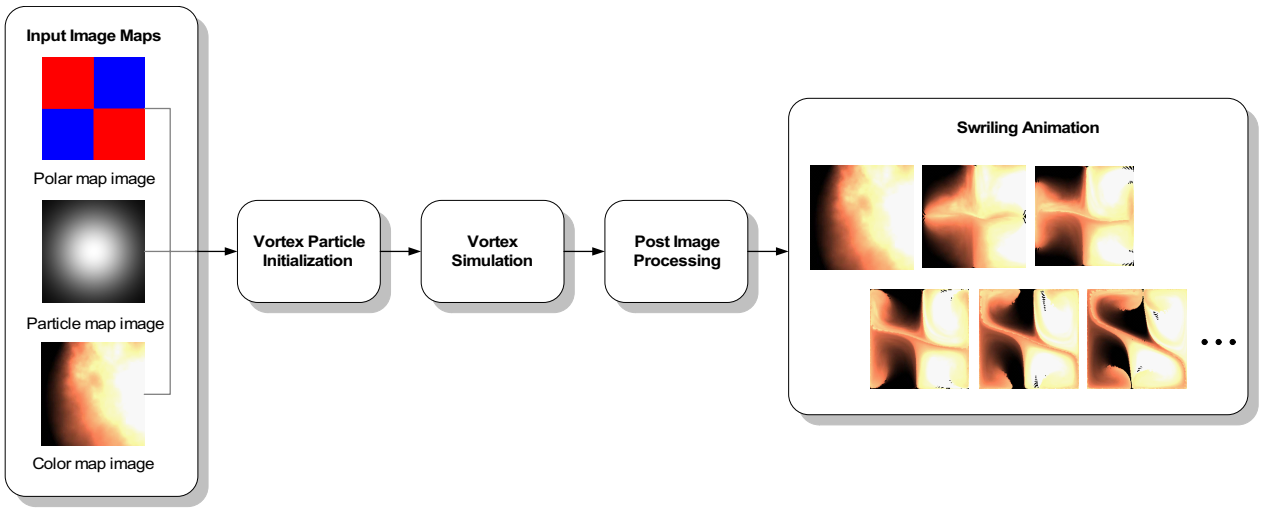

Figure 1. Overview of Swirling Animation

termination of the velocity field of Lagrangian particles. To mitigate the computational expense of vortex methods, we adopt the Vortex-In-Cell (VIC) scheme. This scheme is one of the hybrid vortex methods proposed by Christiansen [3]. In VIC schemes, an Eulerian scheme is used for the efficient computation of the particle velocities in regular bounded domains. The VIC method possesses the following advantages for computer graphics applications:

- It can generate very complex effects with small computing resources.
- It is robust and fast without constraint in the time step for integration.
- It is based on a particle system and hence it can be easily integrated with other particle effects.
- It is very amenable to user interaction.

An overview of our swirling image animation system is shown in Figure 1. The computational grid with the initial vorticles is created from input mapping images. Then, the swirling motion is produced by our implementation of vortex simulation. Finally, every frame image is post-processed either to alleviate aliasing effects or to add other image effects. Many of the resulting images yield compelling effects through very little effort.

Our contributions in this paper include:

1. We propose a physics-based animation system to generate swirling effects in a given image.
2. We introduce a useful formulation of the Navier-Stokes to the graphics community.
3. We show how vortex methods are effectively used to produce complex effects with minimum effort.
4. Additionally, we provide simple interfaces in the form of user-defined image maps to control the resulting effects. Various image attributes are mapped onto various vortex particle properties.

In Section 2 we describe relevant previous work on producing fluid effects on images. Then, in section 3 we describe the vorticity formulation of Navier-Stokes equations and VIC method. The details of implementation are provided in Section 4. We present the results of applying our methods to produce various swirling effects in Section 5. In Section 6 we provide discussion including the possibility of future work.

2. Related Work

We describe pertinent work that has fluid effects and animation as its main goal. Particle system has been the popular procedural techniques for fluid effects [20, 22]. Other work has attempted to blend physics-based techniques with procedural or texture-based methods. Lamorlette and Foster [11] proposed a technique that produces realistic synthesis and animation of flames. Their technique uses physics-based wind fields and Kolmogorov turbulence noise. Similarly, Neyret [14] presented an animated fluid texturing scheme that achieves continuity in space and time and preserves various statistical properties of the given texture. Multiple layers of procedural textures are adaptively advected in a fluid simulation. Then, the multiple textures are blended in frequency space. Finally, small-scale turbulence-like effects are created by adding appropriate procedural noise.

There are techniques that are inspired by flow properties or are completely based on methods from CFD. The work

by Wejchert and Haumann [25] belongs to the former class. The motion of objects in fluid flows are simulated with a set of flow field primitives and boundary layer fluid dynamics around objects. There are two classes of CFD-based simulation techniques used in computer graphics. In the recent past, Eulerian or semi-Lagrangian grid-based methods have successfully used to achieve fluid effects for images [24, 26], rising smoke [8, 23], water motion [7, 9], fire [15], and melting [2]. Witting [26] used compressible fluid formulation derived for a meteorology application to produce two dimensional fluid effects for traditionally-animated films. He also proposed the use of images to control initial conditions for the simulation. However, the control of the simulation is complicated given the large number of parameters employed. In addition, the method has suffered from a severe limitation: the time step often required is very small for the advection phase. The resulting computational expense is often a major drawback of many grid-based methods. Stam's stable fluids method is arguably one of the most popular grid-based techniques for graphics application. It provides a fast and stable solution using semi-implicit numerical scheme at the cost of accuracy. However, the method inherently suffers large numerical dissipation even though Fedkiw et al. [8] proposed the vorticity confinement method to compensate the dissipation by adding artificial vortical force.

Recently, Lagrangian particle-based methods were proposed for interactive fluid animation. Müller et al. [13] proposed an interactive method technique based on smoothed particle hydrodynamics (SPH) to simulate fluids with free surfaces. Premože et al. [18] used the moving particle semi-implict method which is also a Lagrangian method. Although these methods are very similar to vortex methods in terms of numerical schemes employed, they solve the velocity formulation of the Navier-Stokes equations. Vortex methods on the other hand can be very effective for turbulent flow and achieving swirling effects.

The vortex method originated from the work of Rosenhead [21], who evaluated various two-dimensional flows using approximations to the vorticity equations. Restricted by then available computational tools and computer hardware, these methods were often used as numerical models and provided qualitative comparison for solutions of grid-based methods. However, the development of new computational tools and easy availability of powerful hardware has allowed vortex methods to be integrated with many conventional grid-based schemes. As a result, vortex methods are emerging as one of the important methods for the computation of incompressible flows. A useful comparison between grid- and vortex-based methods is provided by Cottet et al. [5]. Vortex methods provide several advantages for computer graphics researchers. They are naturally adapted to complex geometries. In addition, they are free of non-linear convection-related stability conditions, which permits the use of large time steps. Furthermore, one can obtain substantial savings in computational expense over grid-based solvers for a given accuracy. These factors make vortex-based methods better numerical schemes for interactive visual fluid simulation. We will leverage these features of vortex methods to achieve realistic swirling motion in the image plane in this paper.

3. Vortex methods

First, we briefly introduce vorticity formulation of the incompressible and inviscid flow in two dimensions. Then, the VIC numerical scheme is described. The VIC scheme is used to numerically compute the solutions of the vorticity equations. More details can be found in [1, 4, 10]. It should be noted that 2D formulations and solutions suffice for our problem. Our main goal is to produce swirling animations that are initially described on an image grid.

3.1. Vorticity formulation

The Navier-Stokes equations for an incompressible, viscous flow are

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{u}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is fluid velocity, P is pressure, ρ is density, μ is viscosity, and (D/Dt) is the material derivative

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u}\nabla \cdot . \tag{3}$$

The material derivative is interpreted as the time rate of change of a quantity such as temperature or velocity of a material particle. Eq. 1 is an expression of the momentum conservation, while Eq. 2 states the conservation of mass [12].

We define the vorticity vector $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}, \tag{4}$$

By applying the curl operation of Eq. 1, together with a few vector identities, we can get the vorticity transport equation or in essence the vorticity formulation of Eq. 1:

$$\rho \frac{D\boldsymbol{\omega}}{Dt} = \rho\boldsymbol{\omega} \cdot \nabla \mathbf{u} + \mu \nabla^2 \boldsymbol{\omega}. \tag{5}$$

Eq. 5 states that the material derivative $\frac{D\boldsymbol{\omega}}{Dt}$ of vorticity depends on two terms: the rate of deforming vortex lines $\rho\boldsymbol{\omega} \cdot \nabla \mathbf{u}$ and the net rate of vorticity diffusion $\mu \nabla^2 \boldsymbol{\omega}$. Note that the pressure and the associated divergence-free constraint on the velocity have disappeared.

The vorticity equation 5 can be even further simplified for an incompressible and inviscid flow in two dimensions. In the case of two-dimensional flows, the vorticity vector $\boldsymbol{\omega}$ is normal to the velocity $\mathbf{u}$ and the vortex stretching term $\rho\boldsymbol{\omega} \cdot \nabla\mathbf{u}$ vanishes. The vorticity diffusion term $\mu\nabla^2\boldsymbol{\omega}$ disappears for all inviscid flows. Hence, we obtain the following equation

$$\frac{D\boldsymbol{\omega}}{Dt} = 0 \tag{6}$$

for our problem. Note that the vorticity can be treated as a scalar variable in 2D since it can only possess two directions.

The fluid velocity $\mathbf{u}$ can be recovered from the vorticity using the following Poisson equation and velocity-stream function equation

$$\nabla^2\psi = -\boldsymbol{\omega}, \tag{7}$$

$$\mathbf{u} = \nabla \times \psi, \tag{8}$$

where ψ is the stream function which is normal to the plane of motion. This formulation states that the essential physics of the unfolding flow can be described as the time evolution of the vorticity.

3.2. VIC method

We now explain the VIC method to solve the vorticity transportation equations numerically. In vortex methods, the vorticity field is approximated by the distribution of vorticles with compact support. To solve the equations, the position of vorticles is updated by particle trajectory equations. Then, the vorticity of every particle is updated by the vorticity transport equation 5. This Lagrangian approach makes vortex methods robust schemes with minimum dissipation because the problematic nonlinear advection terms are replaced by a set of ordinary differential equations. However, one disadvantage of vortex methods is that the cost of directly computing the vorticity resulting from N vortices is $O(N^2)$. One solution is afforded by hybrid numerical methods that combine vortex methods with Eulerian grid-based schemes. One of the hybrid vortex methods is VIC or Vortex-In-Cell, which was proposed by Christiansen [3]. In the VIC calculations, an Eulerian grid is used in order to compute efficiently the velocity field on the Lagrangian particles. Vorticles are advected by the velocity field. The vorticities of particles are projected on to grid points, and the velocity field is obtained by solving an Poisson's equation on the grid. VIC is simple and easy to implement. In addition, it is fast and accurate enough for visual simulations.

Now, we further describe in detail the VIC method. We denote by the subscript g the grid points and by the subscript p the particle points. Let $\mathbf{x}_g = (i, j)$ be the grid points and $\mathbf{x}_p = (x, y)$ the positions of particles with vorticity values ω_p. Further, let h denote the uniform spacing of the gird.

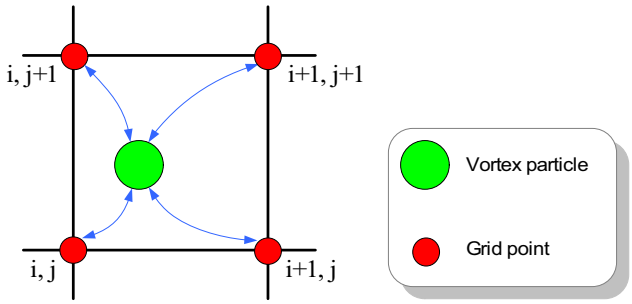

Figure 2. Interpolation between particle and grid points

The vorticity ω_g is obtained from the following assignment:

$$\omega_g = \sum_p \omega_p \phi_g(x_p)\phi_g(y_p) \tag{9}$$

$$\phi_g(x) = \phi\left(\frac{x - x_g}{h}\right) \tag{10}$$

where $\phi(x)$ is the tent filter:

$$\phi(x) = \begin{cases} 1 - |x| & if \quad 0 \le x < 1 \\ 0 & if \quad 1 \le x \end{cases} \tag{11}$$

This method assumes each vorticle to possesses uniform vorticity within a unit square such that the corresponding unit amount of vorticity can be credited to surrounding mesh points through a process of the bilinear interpolation. Conversely, the interpolation of a grid quantity onto particles can be achieved through

$$\omega_p = \sum_g \omega_g \phi_g(x_p)\phi_g(y_p). \tag{12}$$

Figure 2 shows how a particle and its neighbor grid points interact by the assignment Eq. 10 and the interpolation Eq. 12. Thus, the algorithm at every time step is outlined as follows:

1. Assign vorticity to each grid point by summing the individual vorticity contribution from every vorticle using Eq. 10.
2. Solve Eq. 7 to get the stream function ψ.
3. Compute the velocity field $\mathbf{u}$ by differentiating Eq. 8.
4. Interpolate velocity values from the grid to the particles using Eq. 12.
5. Move or advect all the particles by solving the following system of ordinary differential equations

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u} \tag{13}$$

4. Implementation

Two important aspects for all direct numerical simulation methods include numerical discretization and the initial condition. We explain the numerical discretization methods used in our implementation of the VIC method. Later, we describe our methods to initialize the vorticles. Finally, we describe the rendering of particles.

4.1. Numerical discretization

Numerical discretization is needed in VIC methods in three distinct calculations. First, the Poisson equation 7 must be solved to obtain the stream function. We use the standard five-point finite difference approximation to achieve our goal:

$$\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{h^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{h^2} = -\omega_{ij}. \tag{14}$$

The software library *Fishpack* is used to solve this equation with either periodic or constant boundary conditions for the stream function. Second, we use the central difference scheme to compute the velocity using Eq. 8:

$$\begin{aligned} \mathbf{u}[x]_{i,j} &= \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2h} \\ \mathbf{u}[y]_{i,j} &= -\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h}. \end{aligned} \tag{15}$$

The same boundary conditions for the stream function is used for the velocity computation. Finally, to compute the particle trajectories (as derived from Eq.13), the leapfrog scheme is used. One should note that positions and vorticity fields are defined at integral time levels ($t = 0, dt, 2dt \ldots$) and velocities are defined at half-integral time levels ($t = \frac{1}{2}dt, \frac{3}{2}dt \ldots$). The leapfrog difference approximation to Eq. 13 is:

$$\begin{aligned} x_{i,j}^{n+1} &= x_{i,j}^{n} + \mathbf{u}[x]_{i,j}^{n+1/2} dt \\ y_{i,j}^{n+1} &= y_{i,j}^{n} + \mathbf{u}[y]_{i,j}^{n+1/2} dt. \end{aligned} \tag{16}$$

However, we found that the Euler approximation [19] to produce almost the same result that the leapfrog approximation scheme with much faster speed. Hence, we use the Euler scheme in practice.

4.2. Particle initialization

Every vorticle has a color attribute so that color value is advected along the vorticle. The initial vortex particle distribution is determined by three input image maps: *particle*, *color*, and *polarity*. The particle map determines the positions and the vorticities of particles. The color map is used to assign color attributes to the particles, and the polarity map is used to set the sign of the vorticity attribute of the particle. $N_x \times N_y$ computational grid is created from input particle map of the same size. Then, $(N_x-1)\times(N_y-1)$ vorticles are evenly distributed in the center position of the grid with the given stride size.(See Figure 3) The colors and vorticities of the particles are determined from the pixel of the particle map image in the same position. We use the following simple rules to decide the sign of vorticity from a pixel value provided that the range of each color channel of the pixel is between 0 and 1:

- If the intensity of the pixel is greater than a given threshold value, vorticity is positive. Otherwise, vorticity is negative.
- Vorticity is positive at a pixel if the red channel of pixel intensity is non-zero. Otherwise, it is negative if blue channel is non-zero.

By using separate map images for the attributes, more flexible control can be achieved. For example, one image can serve as the source vortex distribution to swirl another image that is provided as a input color map as shown in the fourth row of Figure 4.

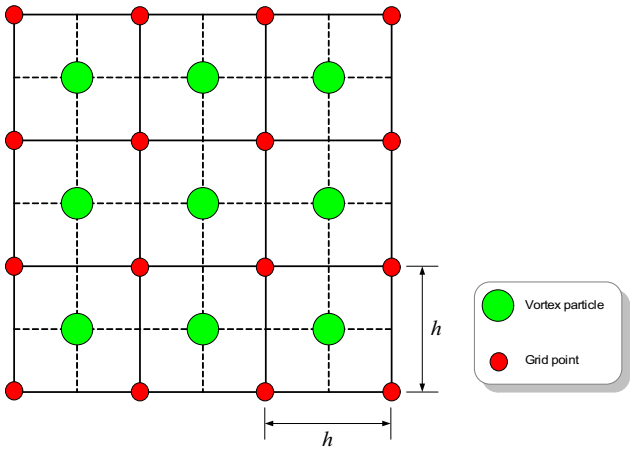

Figure 3. Particle initialization on the computational grid

4.3. Particle rendering and post-processing

A vorticle is rendered as a square pixel primitive with user specified size. Pixel size 2 produces good results in most cases. However, the method suffers an aliasing problem because the initial regular particle mesh structure become distorted as they evolve. We can employ a more ad-

vanced rendering scheme like splatting to correct this problem. Instead, we choose to apply a post-processing filter to the improve the simulated results if needed. In this way, we achieve better performance to allow interactive simulations. Furthermore, this gives users greater flexibility in adding other effects to the results. In practice, a median filter is quite useful to reduce the aliasing of unstructured images such as a glowing fire or flowing water. We found the results are better without filtering for images derived from paintings since filtering often smoothes the details out.

5. Results

We use the Euler method instead of the leapfrog scheme for all the results reported in this Section because it is twice as efficient as the leapfrog scheme and produces similar visual results. Constant boundary conditions along both the x and y directions is used for all the results. The simulation speed is roughly 10 fps for 65000 vorticles. A pixel of size 2 is used to reduce aliasing effects for the results. We used a PC system with Intel Xeon 2GHz processor and 1 GB Ram to generate our results.

Figure 4 shows how the input mapping images work. The first column is an input particle map image for each simulation. The particle map is used to determine the initial vorticle attributes: position and vorticity. The blue and red color denotes the positive vorticity(counter-clockwise rotation) and the negative vorticity(clockwise rotation)respectively. The color map image that is advected by the input particle image is in the second column. In the first row, two groups of vorticles start nearby each other and coalesce eventually. The second row shows a more compelling animation through the use of an explosion color map. The simple swirl motion evolves into an unstable chaotic pattern. The example demonstrates a major advantage of vortex methods – complex swirl patterns can be easily generated with simple image maps and initial conditions. The dune animation illustrates how easily the swirl patterns in an image can be controlled by a particle map. Note that red and blue spots generate very coherent swirl patterns on the corresponding locations. The last example shows how the pepper image is used to advect the flame image. A fluid-like swirling motion is shown. All the images are post-processed with a median filter to reduce the aliasing artifacts.

Figure 5 and 6 show selected frames from the simulated animation for each of the following paintings: Van Gogh's *starry night* and Ando Hiroshige's *Awe Province Navaro Rapids*. The final images are produced without post-processing for these examples. In Figure 5, the local swirling motion is generated around the swirl regions in the original painting. In contrast, the entire image swirls by the initial large rectangular vorticles. These two examples well exemplify the applicability of our method. The animation for every example can be seen in the accompanying Quicktime movie files.

6. Conclusion

We presented a method that uses the point vortex method to produce swirling-like effects in the image plane. The swirling effects are produced by solving two-dimensional incompressible inviscid vorticity transport equations. The vorticity transport equation is better suited to generate realistic fluid motion with small computational expense. Then, the elegant vortex-in-cell(VIC) scheme is used to solve the equations numerically. We also provide an intuitive method to dictate and control the swirling effects through suitable mapping of image properties onto vortex particles. Finally, we demonstrate the potential of our methods by generating various realistic swirling animations from various images.

An immediate problem for our methods are aliasing artifacts arising from a distortion of the particle mesh. A remeshing technique with conservative interpolation schemes can reduce the artifacts. In addition, interactive control would be even more fruitful with an intuitive set of swirl brushes instead of using input images. Furthermore, the automatic generation of a particle map derived from image features would be helpful. For example, an input particle map can be automatically created for the result in Figure 5 by detecting the various swirling regions in the painting. Finally, we plan to extend our work for three dimensional fluid animation.

Acknowledgments

This work is done with the support of Advanced Computing Center for the Arts and Design (ACCAD) fellowship on Human Figure Motion Synthesis, Analysis, and Animation. We would like to thank Dr. David Thompson and Dr. Edward Luke of the Computational Simulation Center at Mississippi State University for bringing the work on vorticity formulation of the Navier-Stokes Equations to our attention .

References

[1] L. an Ying and P. Zhang, editors. *Vortex Methods*. Kluwer Academic Publishers, 1997.

[2] M. Carlson, P. J. Mucha, R. B. Van Horn, III, and G. Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 167–174. ACM Press, 2002.

[3] J. P. Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, 135:189–197, 1997.

[4] G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[5] G.-H. Cottet, B. Michaux, and S. Ossia. A comparison of spectral and vortex methods in threee-dimensional incompressible flows. *Journal of Computational Physics*, 175:702–712, 2002.

[6] D. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, S. Worley, B. Mark, and J. Hart. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 3rd edition, 2002.

[7] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 736–744. ACM Press, 2002.

[8] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press, 2001.

[9] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30. ACM Press, 2001.

[10] K. E. Gustafson and J. A. Sethian, editors. *Vortex methods and Vortex motion*. SIAM, 1991.

[11] A. Lamorlette and N. Foster. Structural modeling of flames for a production environment. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 729–735. ACM Press, 2002.

[12] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics*. Butterworth-Heinemann, 2nd edition, 1997.

[13] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159. Eurographics Association, 2003.

[14] F. Neyret. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–153. Eurographics Association, 2003.

[15] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728. ACM Press, 2002.

[16] K. Perlin. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296. ACM Press, 1985.

[17] K. Perlin and E. M. Hoffert. Hypertexture. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 253–262. ACM Press, 1989.

[18] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. In *Eurographics 2003*, volume 22, pages 401–410, 2003.

[19] W. H. Press, B. P. Flannery, S. A. Teukolosky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[20] W. T. Reeves. Particle systems a technique for modeling a class of fuzzy objects. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 359–375. ACM Press, 1983.

[21] L. Rosenhead. The formation of vortices from a surface of discontinuity. In *Proceedings of the Royal Society of London Serial Archive*, volume 134, pages 170–192, 1931.

[22] K. Sims. Particle animation and rendering using data parallel computation. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 405–413. ACM Press, 1990.

[23] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[24] J. Stam. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.*, 22(3):724–731, 2003.

[25] J. Wejchert and D. Haumann. Animation aerodynamics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 19–22. ACM Press, 1991.

[26] P. Witting. Computational fluid dynamics in a traditional animation environment. *Computer Graphics*, 33(Annual Conference Series):129–136, 1999.

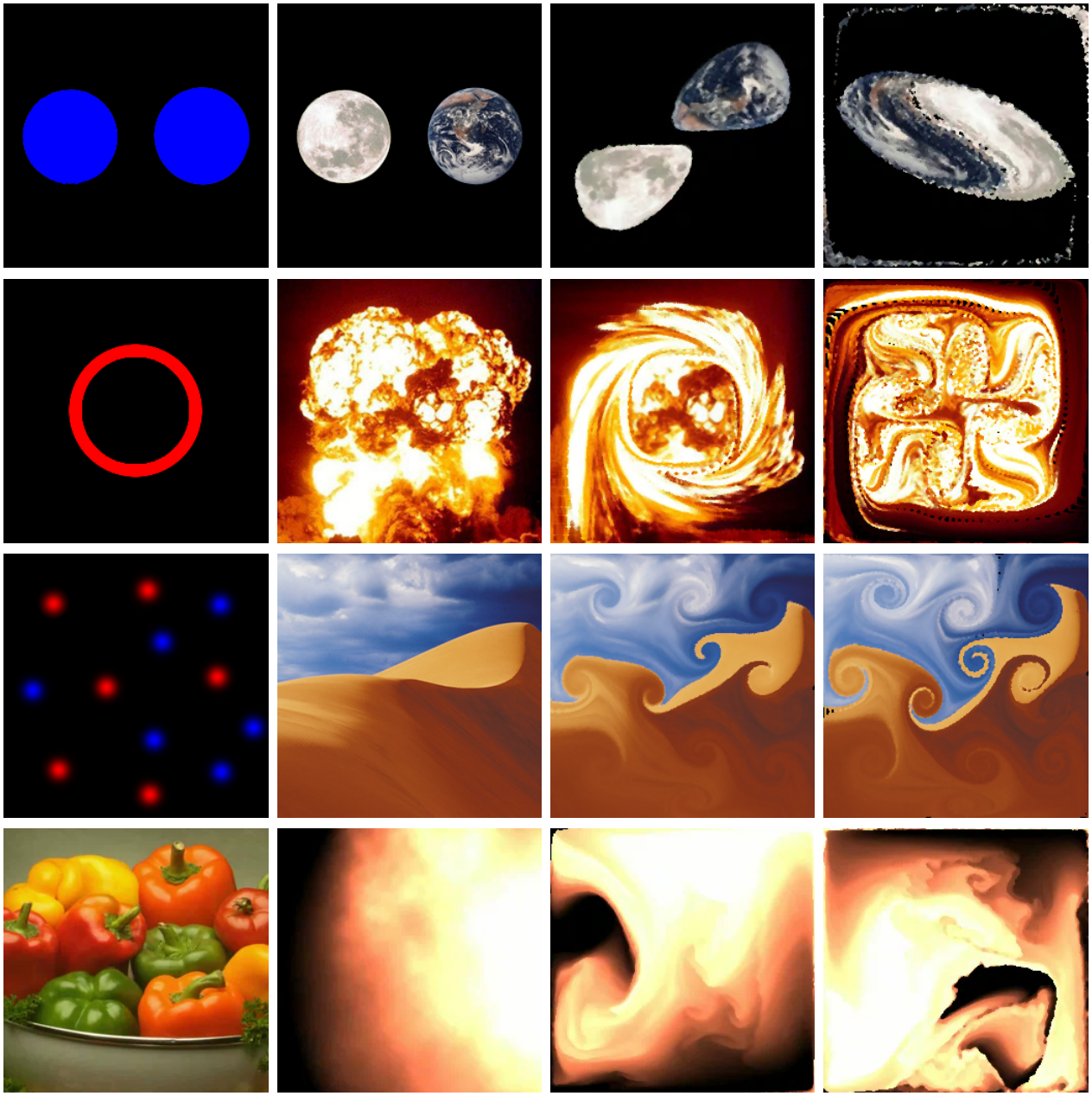

Figure 4. Various particle mapping results: The first column shows input particle map image and the second column shows color map image. The third and fourth column shows two selected frames from the resulting swirling simulation.

Figure 5. Swirly starry night

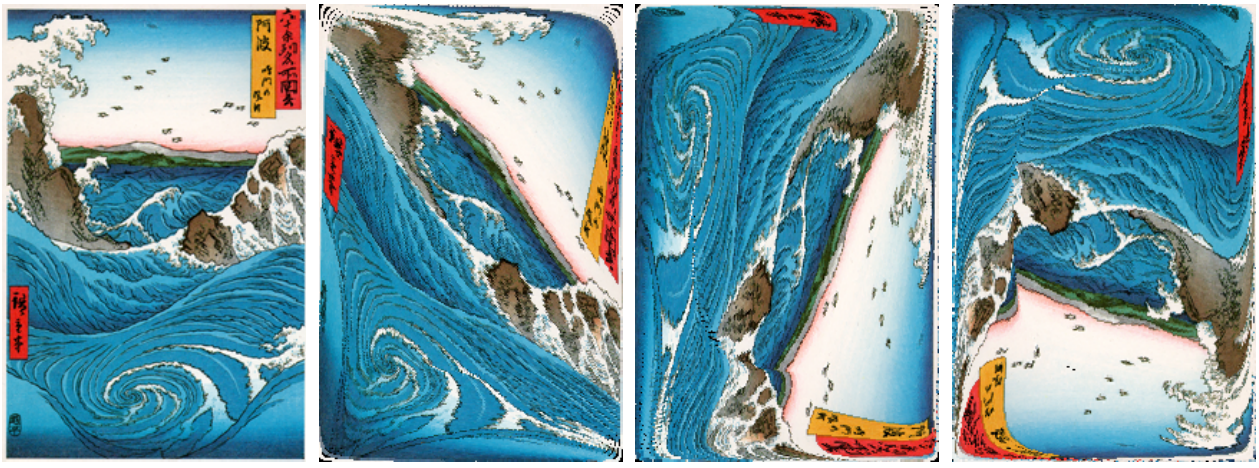

Figure 6. Stormy sea