Pacific Graphics (2011)
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin (Editors)

A Fast Simulation Method Using SPH and Wavelet for Sub-Particle-Scale Turbulent Flow

Makoto Fujisawa[1] and Go Mimura[2] and Toshiyuki Amano[3] and Jun Miyazaki[4] and Hirokazu Kato[4]

[1]University of Tsukuba, Japan
[2]Nintendo Co., Ltd., Japan
[3]Yamagata University, Japan
[4]Nara Institute of Science and Technology, Japan

Abstract

This paper presents a fast simulation method for turbulent flow which uses a particle method and wavelet analysis. To simulate fluid flow, the method uses smoothed particle hydrodynamics (SPH), which discretizes the fluid into a collection of particles, and detects regions where turbulent flow will occur by using wavelet analysis without a spatial grid. By taking the curl of wavelet noise, the turbulent flow is then appended as a divergence-free turbulence velocity field. Additionally, by using vortex subparticles, which characterize the vortex features of turbulence, a subparticle-scale representation of turbulent flow is proposed. Implementing almost all processes on a graphics processing unit (GPU), simulations are performed in near real time.

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

1. Introduction

Fluid simulation is widely being used to model complex fluid phenomena for computer graphics animation. However, most previously developed methods have targeted laminar flow in which the fluid moves in a regular way. Conversely, turbulent flow is unsteady, irregular, and chaotic, and can be observed everyday in our surroundings: for example, smoke from a chimney and water in a river, waterfall, or large ocean wave. Currently, most simulation methods for turbulent flow tend to be time consuming, and the chaotic aspect of turbulence makes simulation difficult. Accordingly, we have developed a fast simulation method for turbulent flow using smoothed particle hydrodynamics (SPH) and a wavelet analysis based on particles.

For computer graphics, turbulent flow is important in enhancing the visual quality of fluid animation and many researchers have incorporated turbulence into existing simulations. Average flow having low-frequency characteristics is generated by solving the Navier-Stokes equations at a coarse resolution, and the smaller details of turbulence are realized by calculating the energy of each manually added vortex or by advecting these vortices. This method can generate much of the fine detail of the flow, even if the size of the detail is smaller than that of the scale of the computational grid. Hence, we can get realistic animation of turbulent flow at low computational cost. However, these simulations still run on offline.

We propose a fast simulation method for turbulent flow which uses a particle method and wavelet analysis. The proposed method uses the particles only to calculate the energy spectrum of the vortices. The turbulent velocity field generated from the wavelet noise is then appended to the base flow. We also make use of the graphics hardware to calculate the wavelet analysis and noise. To represent small-scale vortices, the particles must be small and thus the number of the particles will increase correspondingly. However, van der Laan et al. [vdLGS09] proposed that quasi-turbulence can be added onto a screen space mesh by using a noise function, and this method can render small-scale turbulence without increasing the number of the particles. Here, we take a more physical approach that uses subparticles. The turbulence can be represented as a collection of vortices with a large range of frequencies. A particle is divided into subparticles, which rotate around the original particle to represent a small-scale vortex. In this manner, we can get more realistic results, ow-

DOI: 10.2312/PE/PG/PG2011short/067-072

ing to the technique being able to realize the physical structure of turbulence.

2. Related Works

By using a semi-Lagrangian advection method, Stam [Sta99] developed a solver that is widely used for computer animation of fluids. Although that method can compute the behavior of fluid if a large time-step width is used, the method has a problem with numerical diffusion, which induces the dissipation of turbulence energy, making it difficult to detect the turbulence. BFECC [KLLR05], MacCormack [SFK*08], CIP [KySK08], and adaptive grid [LGF04] methods have been used to try to suppress this dissipation. As an alternative, Fedkiw et al. [FSJ01] regenerated vortices on the grid that were lost in the fluid simulation by using vorticity confinement. However, the vorticity confinement method just enhances the vortices, generating unnecessary turbulence in regions where the fluid is flowing smoothly.

By seeding the vortex particles in regions where turbulent flow will occur, we can append the turbulence to existing fluid simulations. Selle et al. [SRF05] proposed a vortex particle method, where the vortex particle moves in accordance with the vorticity form of the Navier-Stokes equations and transports turbulent energy that is resolved depending on the scale of grid. Pfaff et al. [PTSG09] determined the seed area for the vortex particles from wall-induced turbulence and precomputed boundary layer vorticity around solid objects, sampling the vortex particles in these regions.

A procedural approach that generates small-scale vortices beyond the grid resolution is often used for turbulent flow. Stam [SF93] introduced Kolmogorov theory and developed a procedural turbulence synthesis method to generate the turbulent flow. A divergence-free velocity field, made by taking the curl of a scalar noise field, is used to create a quasi-turbulent flow with a base flow [BHN07]. The technique used by Bridson et al. [BHN07] required specification of the spatial modulation of the turbulence. Pfaff et al. [PTC*10] used energy transport models to determine the distribution of turbulent energy. Zhao et al. [ZYC10] used random forcing to integrate a statistical turbulence field into the simulation. Chang et al. [CKB*10] proposed combining a coarse grid and high-resolution grid for turbulent flow, and Zhu et al. [ZSTB10] used weakly compressible SPH [BT07] and a grid structure to represent turbulent flow induced in solid boundary layers.

Finally, by using wavelet analysis, Kim et al. [KTJG08] determined the region in which the turbulence should be added to the original flow field. They simulated the velocity field for a fluid by using a low resolution grid, based on the Navier-Stokes equations, and synthesized the high resolution turbulence field by using wavelet noise and Kolmogorov theory. The method described here uses similar theory as Kim et al., but we extend their method to particle simulation to achieve fast and accurate computation.

3. Simulation method

The procedure of our method is as follows:

1. Simulate the velocity field from the Navier-Stokes equations using SPH.
2. Calculate the distribution of the turbulence energy via wavelet analysis.
3. Add the divergence-free turbulence field emanating from the wavelet noise.
4. Update the position and velocity of the particles.
5. Update the position and velocity of the vortex subparticles.
6. Extract the mesh of the liquid surface on the basis of the turbulence energy.

This method is easily extendable to other particle methods, for example, the moving particle semi-impact method, because we simply add the turbulence field as an external force.

3.1. Fluid Simulation

We adopt the SPH method to simulate the flow. The Navier-Stokes equations for an incompressible viscous fluid are given by

$$\nabla \cdot u = 0, \tag{1}$$

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

where u, ρ, and p are the velocity, density, and pressure of the fluid, respectively, ν is the kinematic viscosity, and f is the resultant force due to gravity and external forces. The particle method, in which a liquid is represented by particles, is used to discretize the governing equations. SPH is then used to solve the equations. As the number of particles is constant and each particle has constant mass, the total mass of fluid is conserved and we can omit the equation for the conservation of mass (Eq. (1)).

The scalar quantity ϕ is calculated from a weighted sum of neighboring particles:

$$\phi(x) = \sum_{j \in N} m_j \frac{\phi_j}{\rho_j} W(x_j - x, h), \tag{3}$$

where N is the number of neighboring particles, m_j and ρ are the mass and density of particle j, respectively, and the function W is a smoothing kernel with effective radius h. The derivatives of the quantity $\nabla\phi$ are simply evaluated from the derivatives of the kernel function. After the momentum conservation equation (Eq. (2)) is solved via the SPH method, the position and velocity of the particles are updated by using a leapfrog scheme.

3.2. Particle Wavelet Analysis

We generate a turbulent velocity field that consists of a collection of small-scale vortices resulting from Kolmogorov's

law, and small vortices are lost due to numerical diffusion or the Nyquist limit. To generate the small-scale vortices, forward scattering must be reproduced, such that each large vortex is split into two small vortices according to the energy distribution of Kolmogorov's 5/3 spectrum. The vortex energy $\hat{e}$ is evaluated from the spectral component of velocity field $\hat{u}$ in spectral band k:

$$\hat{e}(k) = \frac{1}{2}|\hat{u}(k)|^2. \tag{4}$$

To detect the location at which turbulent flow will occur, we need to consider both the spatial and frequency distribution of the energy (i.e., $\hat{e}(k,x)$).

A short-time Fourier transform (STFT) is one solution to obtain information about both the spatial and frequency domains. To do this the Fourier Transform is applied to a signal that is changed over time or space within a moving window. STFT however suffers from a trade-off between the spatial and frequency resolutions, which are dependent on the size of the window. Kim et al. [KTJG08] solved this problem by using a wavelet transform and then calculating the energy of each grid cell. Our approach is to apply this wavelet transform to the particle method. In the particle method, a grid does not exist that can explicitly define the neighborhood value. Instead, the wavelet transform can be calculated by defining background grid cells that have a projected particle velocity field. However, to use the another grid with almost the same resolution as the particle density would increase the memory usage and computational time and also would cause numerical diffusion when the velocity of the particle is projected onto the grid cell. Therefore, as an alternative, we directly determine the wavelet transform of the velocity field, $\hat{u} = (\hat{u}, \hat{v}, \hat{w})$, and the energy spectrum, $\hat{e}(1/s,x)$, at a scale s by taking a weighted sum of neighboring particles and using the SPH method.

The continuous wavelet transform of the velocity, u, along the x-axis is

$$\hat{u}(s,a,b,c) = \frac{1}{\sqrt{s}} \iiint_{-\infty}^{\infty} u(x)\psi\left(\frac{x-a}{s}, \frac{y-b}{s}, \frac{z-c}{s}\right) dxdydz \tag{5}$$

where s is the wavelet scale, a, b, and c are translational values along each axis, and ψ is the mother wavelet. To determine the neighborhood, a discrete wavelet transform generally assumes that the field is defined at each pixel or upon the grid.

In contrast, the particle method uses a point that is able to move freely along the velocity field. Therefore, we directly calculate the wavelet transform from the particles by using a weighted sum of neighboring particles. The discrete wavelet transform of the velocity, u_i, for particle i is

$$\hat{u}_i = \frac{1}{\sqrt{s}\psi_{sum}} \sum_j u_j \psi\left(\frac{x_i - x_j}{s}, \frac{y_i - y_j}{s}, \frac{z_i - z_j}{s}\right), \tag{6}$$

where $\psi_{sum} = \sum_j \psi$. If a grid is used, the number of particles around an individual cell is constant. Conversely, when the particle method is used, the number of neighboring particles will vary according to the cell's position and time, so that the absolute value of wavelet transform tends to be large near the surface of a liquid. This is because there is only a small number of particles at the liquid ' s surface and the influence of a particle will therefore be strong. To prevent this, we introduce ψ_{sum} to normalize the value. The energy of vortex $e_i(k)$ can be calculated by substituting $\hat{u}$ into Eq. (4).

Figure 1 shows the energy spectrum calculated from Eq. (5) by using a background grid and by using our method described above. It can be seen in Fig. 1 that the spectrum resulting from our method is almost the same as that generated by projecting the velocity of particles onto a background grid.

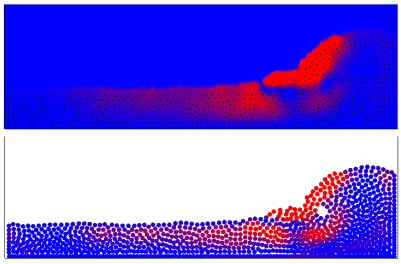


Figure 1: *Energy spectrum calculated using a grid (top) and particles (bottom). High and low energies are shown in red and blue, respectively.*

3.3. Wavelet Turbulence Synthesis

tjTo simulate the forward scattering of the vortices, Kolmogorov's theory is used. The energy spectrum of turbulence in frequency domain is

$$\hat{e}(k) = \alpha\varepsilon^{\frac{2}{3}} k^{-\frac{5}{3}}, \tag{7}$$

where α is the Kolmogorov constant and ε is the mean energy dissipation rate per unit mass. As indicated by Kim et al. [KTJG08], Eq. (7) can be rewritten and the wavelet turbulence function is

$$y(x) = \sum_{i=i_{min}}^{i_{max}} w(2^i x) 2^{-\frac{5}{6}(i-i_{min})}, \tag{8}$$

where $[i_{min}, i_{max}]$ is the bandwidth of the spectrum and $w(x)$ is the divergence-free velocity field [BHN07] generated from the wavelet noise [CD05]. We want to generate the particle-scale turbulence and, to this end, i_{max} can be the resolution such that the width of a cell is the same as the particle diameter. i_{min} is determined based on the wavelet scale, s.

The turbulent force from the energy $\hat{e}_i(k)$ and wavelet function is

$$f_i^{turb} = A\frac{\rho_i}{\Delta t}\hat{e}_i(k)y(x_i), \tag{9}$$

where A is a user-controlled parameter that changes the scale of the turbulence. The turbulent force is added as the external force term in equation (2).

3.4. Sub-Particle-Scale Turbulence

Turbulent flow can be captured as a collection of vortices with a large range of frequencies. To completely represent the flow, we must handle the full range of vortices. Unfortunately, the scale of vortices that can be represented within the simulation depends on the scale of the particles. To simulate small-scale vortices, it is necessary to use smaller particles; this in turn increases the number of particles and the computational time. We use a small particle, named a vortex subparticle to represent sub-particle-scale turbulence. This is similar to the vortex particle method [SRF05], but here the sub-particles only rotate around the particles used in SPH and depend on the turbulence energy. As a result, the sub-particles hardly influence the computation time because they do not use the neighboring particles. Figure 2 illustrates the concept of vortex sub-particles. The SPH particles, p_i^0, can be divided into pairs of vortex sub-particles, $p_i^{1,j}$ $j = 0, 1$ and the sub-particles can also be divided recursively. The superscript indicates the level of the subparticle (i.e., Level 0 represents the SPH particles).

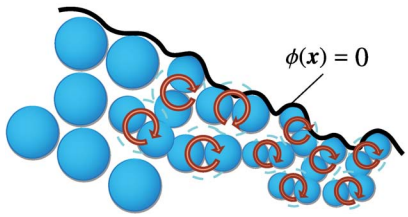


Figure 2: *Small-scale vortex sub-particles.*

Figure 3 shows the relationship between particles p^L and p^{L+1}. $Vc_i^{L,j}$ is a unit vector from $p_i^{L,j}$ to $p_i^{L+1,2j}$, where $j = 0, 1, ..., 2^L - 1$. The radius of $p_i^{L+1,j}$ is $r_L = 2^{-L/3}r_0$, where r_0 is the radius of SPH particle p^0. The total volume of two sub-particles must be equal to that of the SPH particle.

The sub-particles are advected in the same manner as their parent particle and are rotated by the energy of the turbulent flow. The energy of each particle, p^0, as calculated from Eq. (7), is cascaded down to its sub-particles according to Kolmogorov's theory, and represents the forward scattering of the vortices. Each subparticle has rotation axis, $a_i^{L,j}$ ($\perp c_i^{L,j}$), and the angular velocity, ω, is calculated from

$$|\hat{u}(\frac{1}{2r_{L+1}}, x_i)| = \sqrt{2\hat{e}(\frac{1}{2r_{L+1}}, x_i)} = \sqrt{2\hat{e}(\frac{1}{2r_0}, x_i)(2^{-\frac{5}{9}})^{L_i}}, \tag{10}$$

$$\omega_i^{L,j} = \frac{|\hat{u}(\frac{1}{2r_{L+1}}, x_i)|\Delta t}{r_{L+1}}. \tag{11}$$

To reconstruct the features of the turbulence, the axis $a_i^{L,j}$ must be randomly rotated.

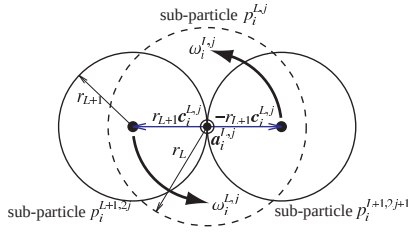


Figure 3: *Relationship between particle p^L and p^{L+1}*

3.5. Surface Extraction

The surface of the liquid is defined as the zero level set of the following function:

$$\phi(x) = \sum_i^N W_{sub}(x_i, L_i), \tag{12}$$

where W_{sub} is a kernel function from which the level of sub-particle used in the calculations is chosen, depending on the energy spectrum. We define a criterion for the energy value, e_{cri}, and the level, L_i, of the subparticle for the kernel must satisfy

$$e_{cri} = \hat{e}(\frac{1}{2r_L}, x)(2^{-\frac{5}{9}})^{L_i}, \tag{13}$$

where $\hat{e}(s, x)$ is the energy spectrum at x with scale s. The number of levels of division for a particle with high energy will be greater than that for a particle with low energy, so that small vortices appear in only regions where the energy of the turbulence is large. W_{sub} is defined by the level, L_i, and a user defined maximum, L_{max}:

$$W_{sub}(x_i, L_i) = \begin{cases} W(x_i, r_0) & L_i \le 0, \\ \sum_{j=0}^{2^{L_{max}}} W(x_i^{L_{max},j}, r_{L_{max}}) & L_i \ge L_{max}, \\ (L_i^{up} - L_i)\sum_{j=0}^{2^{L_i^{down}}} W(x_i^{L_i^{down},j}, r_{L_i^{down}}) & \\ \quad +(L_i - L_i^{down})\sum_{j=0}^{2^{L_i^{up}}} W(x_i^{L_i^{up},j}, r_{L_i^{up}}) & \text{otherwise}, \end{cases} \tag{14}$$

where $L_i^{down} = \lfloor L_i \rfloor$ and $L_i^{up} = \lfloor L_i \rfloor + 1$. We then extract a triangle mesh, corresponding to the liquid surface, using the marching cubes algorithm [LC87].

4. Results

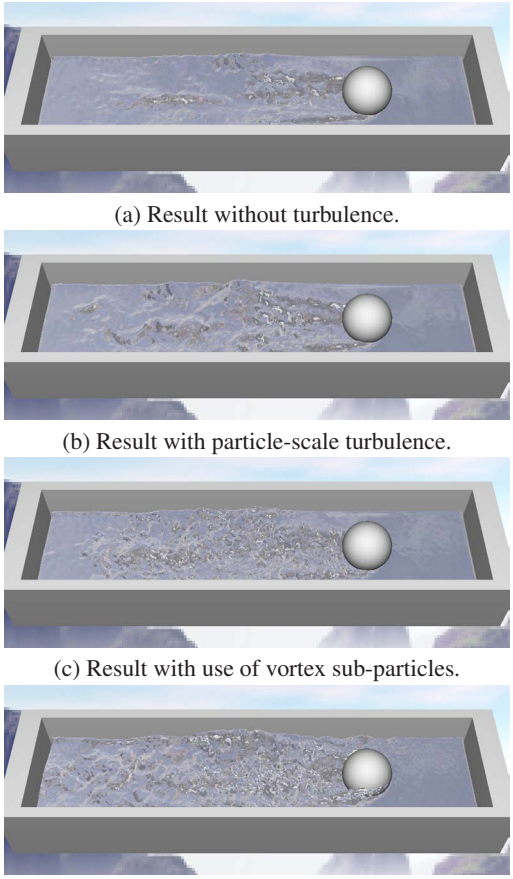
This section describes the results of applying the proposed method to several test cases. All results were generated on a computer equipped with a 2.93 GHz Intel Core i7 Duo CPU and an NVIDIA GeForce GTX580 GPU. The algorithm was predominantly implemented on a GPU by using the NVIDIA CUDA architecture, and a Mexican hat wavelet was used as the mother wavelet.

Figure 5 shows flooding in a valley. Since our method can simulate turbulence, it can be seen that the flow is disturbed. The maximum number of fluid particles in the scene was 40,000, and the computation time for the simulation was approximately 15 ms per step. The calculation of the energy spectrum and wavelet turbulence required about 70% of the simulation time. We used a marching cubes method to construct the liquid surface mesh. Mesh construction required only about 15 ms with, at most, 80,000 triangles to generate (a) and (b). We also implemented the marching cubes algorithm in parallel on the GPU. Figure 5(c) shows the output with subparticle-scale turbulence included; the total computational time was about 120 ms per step. The computation time for the simulation was almost the same as for (b), but the calculation of Eq. (12) took a greater amount of time because the particles for meshing include the vortex sub-particles.

Finally, we verified the effect of the vortex sub-particles. Figure 4 shows the trailing vortex left by a moving ball. The number of particles is approximately 23,000 for Fig. 4(a), (b), and (c), and 184,000 for (d). For comparison, we ran the simulation with a large number of particles. The maximum level of sub-particle division was 3, such that the resolution of particles for rendering in Fig. 4 (c) was same as that in (d). It is seen that our method can generate very small turbulent flows, despite the number of particle being relatively small. However, an artifact of our method is that the sub-particle-scale vortices remained longer than those for (d), resulting in an unnatural animation. Hence, it is concluded that, to generate a more realistic turbulent flow, the vortex sub-particles should simulate not only the forward scattering of vortices, but also the backward scattering or dissipation due to the viscosity of the fluid.

5. Conclusions

We have presented a method to simulate turbulent flow using an SPH method and wavelet analysis. Our method directly calculates the turbulence energy by taking a weighted sum of neighboring particles and adds the turbulence field as a external force. This enables fast simulations that include various turbulent flows. In addition, we synthesized particle-scale vortices—which are dissipated by numerical diffusion— from the energy spectrum by using a wavelet turbulent noise based on Kolmogorov's theory and sub-particle-scale vortices created from sub-particle splitting. Moreover, we performed the simulations in near real time by using a GPU in parallel.



(a) Result without turbulence.

(b) Result with particle-scale turbulence.

(c) Result with use of vortex sub-particles.

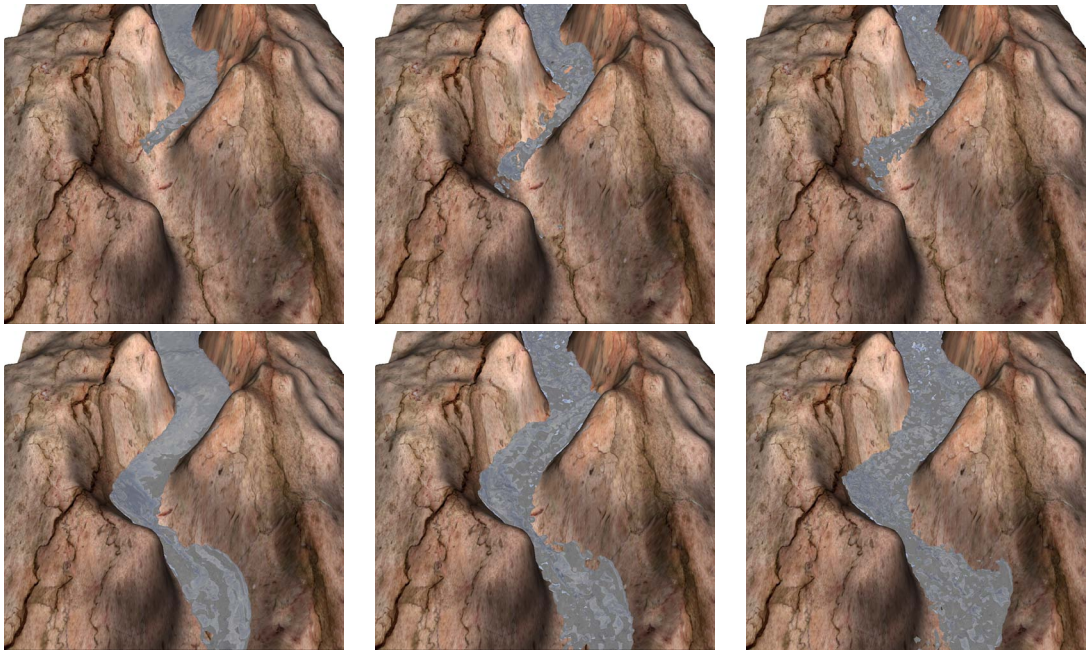
(d) Result with particle-scale turbulence and 184,000 particles.

Figure 4: *Comparison using vortex sub-particles.*

Although our method could successfully realize small vortices by simulating the forward scattering via particle splitting, the lack of backward scattering gave rise to some lingering visual artifacts due to the residual energy. For realistic rendering, bubbles, foam, and splashes are also vital, and we will be able to generate these effects based on the turbulence energy. In addition, to speed up the calculations further, we can use a more efficient algorithm for the surface extraction, such as a screen space mesh [MSD07], and utilize the positional relationship between the sub-particles to accelerate the calculation of Eq. (12). For a liquid simulation, the sub-particle near the surface only affects the final rendering. It might save a lot of computation if we eliminate the sub-particles that are not near the surface.

References

[BHN07] BRIDSON R., HOURIHAM J., NORDENSTAM M.: Curl-noise for procedural fluid flow. In *Proc. SIGGRAPH 2007* (2007), p. 46. 2, 3



(a) No turbulence. (b) Particle-scale turbulence. (c) Subparticle-scale. turbulence

Figure 5: *Flooding in a valley.*

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 209–217. 2

[CD05] COOK R. L., DEROSE T.: Wavelet noise. In *Proc. SIGGRAPH 2005* (2005), pp. 803–811. 3

[CKB*10] CHANG T., KIM H., BAE J., SEO J., NOH J.: Multilevel vorticity confinement for water turbulence simulation. In *Proceedings of CGI 2010 (The Visual Computer)* (2010). 2

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. In *Proc. SIGGRAPH 2001* (2001), pp. 15–22. 2

[KLLR05] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Flowfixer: Using bfecc for fluid simulation. In *Eurographics Workshop on Natural Phenomena* (2005). 2

[KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. In *Proc. SIGGRAPH 2008* (2008), pp. 1–6. 2, 3

[KySK08] KIM D., YOUNG SONG O., KO H.-S.: A semi-lagrangian cip fluid solver without dimensional splitting. *Computer Graphics Forum (Proc. Eurographics) 27*, 2 (2008), 467–475. 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings) 21*, 4 (1987), 163–169. 4

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *Proceedings of SIGGRAPH 2004* (2004), pp. 457–462. 2

[MSD07] MÜLLER M., SCHIRM S., DUTHALER S.: Screen space meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 9–15. 5

[PTC*10] PFAFF T., THUEREY N., COHEN J., TARIQ S., GROSS M.: Scalable fluid simulation using anisotropic turbulence particles. In *Proceedings of SIGGRAPH Asia 2010* (2010), pp. 174:1–174:8. 2

[PTSG09] PFAFF T., THUEREY N., SELLE A., GROSS M.: Synthetic turbulence using artificial boundary layers. In *Proceedings of SIGGRAPH Asia 2009* (2009), pp. 1–10. 2

[SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Proc. SIGGRAPH1993* (1993), pp. 369–376. 2

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable maccormack method. *J. Sci. Comput. 35*, 2-3 (2008), 350–371. 2

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. In *Proc. SIGGRAPH 2005* (2005), pp. 910–914. 2, 4

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

[vdLGS09] VAN DER LAAN W. J., GREEN S., SAINZ M.: Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (2009), pp. 91–98. 1

[ZSTB10] ZHU Y., SIFAKIS E., TERAN J., BRANDT A.: An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2 (2010), 1–18. 2

[ZYC10] ZHAO Y., YUAN Z., CHEN F.: Enhancing fluid animation with adaptive, controllable and intermittent turbulence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2010). 2