Turbulent Wind Fields for Gaseous Phenomena

Jos Stam
Eugene Fiume[0]

Department of Computer Science
University of Toronto
10 King's College Circle
Toronto, Canada, M5S 1A4

Abstract

The realistic depiction of smoke, steam, mist and water reacting to a turbulent field such as wind is an attractive and challenging problem. Its solution requires interlocking models for turbulent fields, gaseous flow, and realistic illumination. We present a model for turbulent wind flow having a deterministic component to specify large-scale behaviour, and a stochastic component to model turbulent small-scale behaviour. The small-scale component is generated using space-time Fourier synthesis. Turbulent wind fields can be superposed interactively to create subtle behaviour. An advection-diffusion model is used to animate particle-based gaseous phenomena embedded in a wind field, and we derive an efficient physically-based illumination model for rendering the system. Because the number of particles can be quite large, we present a clustering algorithm for efficient animation and rendering.

CR Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.3.3 [**Computer Graphics**]: Picture/Image Generation; G.3 [**Probability and Statistics**]: Probabilistic algorithms.
Additional keywords and phrases: turbulent flow, stochastic modelling, Kolmogorov energy spectrum and cascade, transport model of illumination, Fourier synthesis, advection-diffusion, gaseous phenomena.

1 Introduction

We have come to appreciate the central role that irregularity plays in modelling the shape of natural objects. The analogue for wind and fluids is *turbulence*, and its effects are no less essential to the realistic portrayal of gaseous natural phenomena: curling wisps of smoke, mist blowing across a field, car exhaust, an aerosol spray, steam rising from a coffee mug, clouds forming and moving across the sky, the fall of leaves, a swirl of dust in a room, a hurricane. These effects are caused by the interaction of objects with a wind velocity field. Modelling the effect of wind requires that we model both the wind field and this interaction. Both Sims [14] and Wejchert and Haumann [17] model a wind field as the superposition of deterministic fields. Modelling a visually convincing turbulent wind field this way is painstaking. The greatest success in this direction was the particle-based "Blowing in the Wind" animation by Reeves and Blau [10].

Stochastic modelling is a natural alternative strategy. In [13], Shinya and Fournier describe an approach developed independently of ours but which has some similarities. They employ stochastic processes and Fourier synthesis to derive a wind field in spatiotemporal frequency domain, and invert the result to get a periodic space-time wind field. We employ the same paradigm, but our model and application are quite different. Although both wind models can be applied to a wide range of phenomena, and [13] demonstrates this very well, their main concern is with coupling the wind model to macroscopic physical models of rigid or deformable objects, whereas we are mostly concerned with microscopic interaction with gaseous and fluid phenomena. Consequently, our model of turbulence is dissimilar: Shinya and Fournier assume a constant deterministic temporal evolution (Taylor Hypothesis), while for us temporal evolution is also a stochastic process. Our wind model also differs in that an animator has direct control over deterministic and stochastic components of a field.

In this paper, turbulent wind fields are modelled as stochastic processes. The model is empirically plausible[5]. A wind field is generated from large-scale motion and from the statistical characteristics of the small turbulent motion, both freely chosen by an animator. This is analogous to modelling rough terrain by providing the global shape as given by a set of height samples, and the desired roughness of the terrain [2]. The large scale of the wind field will be modelled using simple wind field primitives [14, 17]. The small scale of the wind field will be modelled as a three-dimensional random vector field varying over space and time. This field is generated using inverse an FFT method[16] that we have generalized to a vector field. The resulting wind field has two desirable properties. First, it is periodic and is thus defined for any point in space-time. Second, it is generated on a discrete lattice and can be interactively calculated using four-linear interpolation.

Gases have been modelled in several ways. Ebert models a gas as a solid texture. With some trial-and-error (and in our experience, significant human effort), realistic animations were obtained[1]. Sakas models a gas as a 3-D random density field, generating it using spectral synthe-

[0]The financial support of the Natural Sciences and Engineering Research Council of Canada and of the Information Technology Research Centre of Ontario is gratefully acknowledged. The helpful suggestions of the referees are greatly appreciated.

sis [12]. While spectral synthesis is useful in generating turbulent wind fields, it is not ideal for directly generating density fields: visual artifacts appear due to the periodicity of the field and the entire density field must be computed at once. The temporal evolution of the density field is limited to simple translations. Both of the above models are computationally expensive to visualize, and hence interactive modelling is not feasible. Using physically-based turbulence to animate density fields is mathematically nontrivial, but we shall show that this can be done efficiently.

We model gases as density distributions of particles. The evolution of a density distribution within our wind field is described by an advection-diffusion equation. We efficiently solve this equation by modelling the gas as a "fuzzy blobby" with time varying parameters. A fast ray-tracing algorithm is used, based on a front to back single-scattering illumination model, to render such a density distribution.

2 A Multiple-Scale Wind Field Model

Physically, wind fields are the result of the variations of the velocity $\mathbf{u}(\mathbf{x}, t)$ and the pressure $p(\mathbf{x}, t)$ of a fluid (including air) over space and time. These variations are caused by various forces: external forces $\mathbf{F}$ applied to the fluid, non-linear interactions between different modes of the velocity field and viscous dissipation at a rate ν. By summing these forces and equating them to the acceleration of the fluid we obtain the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho_f}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{F}, \qquad (1)$$

where ρ_f is the density of the fluid. If the velocities of the fluid are much smaller than the speed of sound, we can assume that the fluid is incompressible [5], i.e.,

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

When proper initial conditions and boundary conditions are specified, Eqs. 1 and 2 are sufficient to solve for the velocity field and the pressure of the fluid for any time instant.

The above equations could be used to animate realistic wind fields. One would first specify the physical properties of the fluid that make up the model, including an initial velocity field and boundary conditions. One would then control the fluid motion by applying external forces. Realistic wind fields would be obtained by solving the Navier-Stokes equations as needed. This is entirely akin to the control problem for articulated figures, and it shares the same difficulties. First, a desired effect is hard to achieve by "programming" it using only external forces. Second, the non-linearities present in the Navier-Stokes equations make them hard to solve numerically, especially in the presence of turbulence (low viscosity). Linearizing the equations can improve stability and efficiency, which has been done by Kass and Miller to model the surface of water [4]. This results in highly viscous fluids that do not exhibit turbulence.

We shall model a turbulent wind field by separating it into a large-scale component $\mathbf{u}_l$ and a small scale component $\mathbf{u}_s$. The large-scale term is composed of simple wind fields, resulting in very viscous fluids. The small-scale term is a random field. We shall make a useful but physically implausible assumption that the components are independent, that is, that large scales do not affect the small scales and vice-versa. Hence we will write

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_l(\mathbf{x}, t) + \mathbf{u}_s(\mathbf{x}, t). \qquad (3)$$

This assumption permits the real-time simulation and independent control of both large-scale and small-scale effects. The results, as we shall see, are quite convincing. We shall further discuss this assumption in our conclusions.

3 Small Scale Modelling

3.1 Random Vector Fields

In this section we will denote the small scale component $\mathbf{u}_s$ simply by $\mathbf{u}$. It is defined as a random space-time vector field, a function that assigns a random velocity to each point $(\mathbf{x}, t)$ in space-time [15]. We shall invoke the standard Gaussian assumption [7]: that the random vector field is entirely determined by its second-order moments. These moments are obtained by statistically averaging (denoted by $\langle\ \rangle$) components of the evolving random velocity field. We will assume that the mean values of each component $\mu_i(\mathbf{x}, t) = \langle u_i(\mathbf{x}, t)\rangle$ $(i = 1, 2, 3)$ of $\mathbf{u}$ are constant and equal to zero. The *cross-correlation* between different components of the velocity field at two different points in space-time $(\mathbf{x}, t)$ and $(\mathbf{x}', t')$ are given by the functions

$$\Gamma_{ij}(\mathbf{x}, t; \mathbf{x}', t') = \frac{\langle u_i(\mathbf{x}, t) u_j(\mathbf{x}', t')\rangle}{\langle \mathbf{u}^2\rangle}, \quad i, j = 1, 2, 3. \qquad (4)$$

Where $\langle \mathbf{u}^2\rangle = \langle u_1^2 + u_2^2 + u_3^2\rangle$ denotes the variance of the velocity field and physically is equal to twice the kinetic energy of the field. We will assume that the velocity field is *homogeneous* in space and *stationary* in time, which means that the cross-correlation only depends on the difference $\mathbf{r} = \mathbf{x}' - \mathbf{x}$ between the two points and the difference $\tau = t' - t$ between the two times: $\Gamma_{ij}(\mathbf{x}, t; \mathbf{x}', t') = \Gamma_{ij}(\mathbf{r}, \tau)$.

Homogeneous velocity fields have a corresponding representation in spatial-frequency domain via a spatial Fourier transform. Intuitively this transformation can be thought of as a decomposition of the velocity field into "eddies" of different sizes: large eddies correspond to small spatial frequencies and conversely for small eddies. The stationarity of the velocity field allows it to be represented in frequency domain by a temporal Fourier transform. We will denote spatial frequencies by $\mathbf{k} = (k_1, k_2, k_3)$ and temporal frequencies by ω.[1] We represent the velocity field in frequency domain via the usual Fourier transform:

$$\hat{\mathbf{u}}(\mathbf{k}, \omega) = \int\int \mathbf{u}(\mathbf{x}, t) \exp(-i\mathbf{k} \cdot \mathbf{x} - i\omega t)\, d\mathbf{x}dt. \qquad (5)$$

Writing the transform in this manner facilitates its separation into spatial and temporal frequency components. The Fourier-domain equivalent of the cross-correlation functions are the cross-spectral density functions:

$$\Phi_{ij}(\mathbf{k}, \omega) = \langle \hat{u}_i^*(\mathbf{k}, \omega)\hat{u}_j(\mathbf{k}, \omega)\rangle, \quad i, j = 1, 2, 3, \qquad (6)$$

where the "$*$" denotes the complex conjugation. Conveniently for us, the cross-spectral density functions and the cross-correlation functions are Fourier-transform pairs [15].

Finally, we assume that the velocity field is spatially *isotropic*, meaning that the cross-correlation functions are invariant under rotations. Thus the cross-correlation functions only depend on the distance $r = \|\mathbf{r}\|$ between two points. Isotropy and incompressibility (Eq. 2) imply that the cross-spectral density functions are of the form [5]

$$\Phi_{ij}(\mathbf{k}, \omega) = \frac{E(k, \omega)}{4\pi k^4}(k^2\delta_{ij} - k_i k_j), \quad i, j = 1, 2, 3, \qquad (7)$$

[1] In the turbulence literature, the term *wave number* is often used instead of *spatial frequency*. We will use *spatial frequency*, which is more common in computer graphics, but we shall denote spatial frequencies by $\mathbf{k}$, reserving the letter ω for temporal frequencies.

where δ_{ij} is the *Kronecker delta*, k is the length of the spatial frequency $\mathbf{k}$ and E is a positive function called the *energy spectrum function*. Its physical interpretation is that it gives the contribution of all spatial frequencies of length k and frequency ω to the total kinetic energy of the velocity field:

$$\frac{1}{2}\langle \mathbf{u}^2 \rangle = \int_0^\infty \int_{-\infty}^{\infty} E(k,\omega)\, d\omega\, dk. \tag{8}$$

3.2 The Energy Spectrum Function

Eq. 7 states that the structure of a velocity field (via its cross-spectral density functions) is entirely determined by its energy spectrum function. In other words, an animator can control the qualities of turbulent motion by specifying the shape of the energy spectrum. This function can be arbitrary as long as the integral of Eq. 8 exists. In the turbulence literature one can find a wide variety of different energy spectra for various phenomena. These models are either determined from experimental data or obtained from simplifying assumptions about the fluid. The best-known example of the latter for turbulence that has reached a steady-state (i.e., $\int_{-\infty}^{\infty} E(k,\omega)\,d\omega \rightarrow E(k)$) is the *Kolmogorov energy spectrum* [5]:

$$E_K(k) = \begin{cases} 0 & \text{if } k < k_{\text{inertial}} \\ 1.5\ \epsilon^{3/2}\ k^{-5/2} & \text{otherwise} \end{cases} \tag{9}$$

This spectrum results from an *energy cascade*, where energy introduced at frequency k_{inertial} is propagated to higher frequencies at a constant rate ϵ. Instead of invoking Taylor's Hypothesis [13] we model the temporal frequency dependence of the energy spectrum function $E(k,\omega)$ by multiplying the Kolmogorov energy spectrum $E_K(k)$ by a temporal spread function $G_k(\omega)$ subject to:

$$\int_{-\infty}^{\infty} E(k,\omega)\,d\omega = E_K(k)\int_{-\infty}^{\infty} G_k(\omega)\,d\omega = E_K(k). \tag{10}$$

This guarantees conservation of kinetic energy (cf. Eq. 8). Furthermore, we want the small eddies to be less correlated in time than the large eddies. Spatially, this means that small eddies spin, ebb and flow more quickly than large eddies; this behaviour can be observed when watching a water stream or smoke rising from a cigarette. We can achieve this behaviour by setting G_k to a Gaussian with a standard deviation proportional to k:

$$G_k(\omega) = \frac{1}{\sqrt{2\pi}\,k\sigma}\exp\left(-\frac{\omega^2}{2k^2\sigma^2}\right). \tag{11}$$

Indeed, for large eddies (as $k \rightarrow 0$), G_k is a spike at the origin, corresponding to the spectral distribution of a highly-correlated signal; for small eddies (as $k \rightarrow \infty$) the spectral density becomes constant, denoting an uncorrelated signal.

3.3 Generating the Small Scale Component

We now describe an algorithm to generate a random velocity field having specified cross-spectral density functions Φ_{ij}. The algorithm is a generalization of Voss's inverse FFT method[16]. The idea is to filter an uncorrelated white noise velocity field in the Fourier domain, and then to take an inverse Fourier transform to obtain the desired random velocity field. The challenge is thus to find the right filter such that the resulting velocity field has the desired statistics.

We first compute the velocity field in the frequency domain for discrete spatial frequencies (i,j,k) and temporal frequencies l.[2] Let us assume that the discretization is uniform and that there are N samples per dimension. Then the discrete Fourier transform (DFT) of the velocity field $\hat{\mathbf{u}}_{i,j,k,l}$ is defined on a discrete lattice of size $3N^4$. To ensure that the resulting space-time velocity field is real valued, the elements of the DFT must satisfy the following symmetries: $\hat{\mathbf{u}}_{i,j,k,l} = \hat{\mathbf{u}}^*_{N-i,N-j,N-k,N-l}$, where the indices are taken modulo N, i.e., $N - 0 = 0$[9]. In the special case when the indices on both sides of the equality are identical (e.g., $\hat{\mathbf{u}}_{N/2,0,N/2,N/2}$) we have to set the imaginary parts of $\hat{\mathbf{u}}_{i,j,k,l}$ to zero. The following algorithm generates a DFT with the required properties.

for i,j,k,l in $\{0,\ldots,N/2\}$ do
    compute $\hat{\mathbf{u}}_{i,j,k,l}$, $\hat{\mathbf{u}}_{N-i,j,k,l}$, $\hat{\mathbf{u}}_{i,N-j,k,l}$, $\hat{\mathbf{u}}_{i,j,N-k,l}$,
        $\hat{\mathbf{u}}_{i,j,k,N-l}$, $\hat{\mathbf{u}}_{N-i,N-j,k,l}$, $\hat{\mathbf{u}}_{N-i,j,N-k,l}$, $\hat{\mathbf{u}}_{N-i,j,k,N-l}$
    $\hat{\mathbf{u}}_{N-i,N-j,N-k,N-l} = \hat{\mathbf{u}}^*_{i,j,k,l}$
    $\hat{\mathbf{u}}_{i,N-j,N-k,N-l} = \hat{\mathbf{u}}^*_{N-i,j,k,l}$
    $\hat{\mathbf{u}}_{N-i,j,N-k,N-l} = \hat{\mathbf{u}}^*_{i,N-j,k,l}$
    $\hat{\mathbf{u}}_{N-i,N-j,k,N-l} = \hat{\mathbf{u}}^*_{i,j,N-k,l}$
    $\hat{\mathbf{u}}_{N-i,N-j,N-k,l} = \hat{\mathbf{u}}^*_{i,j,k,N-l}$
    $\hat{\mathbf{u}}_{i,j,N-k,N-l} = \hat{\mathbf{u}}^*_{N-i,N-j,k,l}$
    $\hat{\mathbf{u}}_{i,N-j,k,N-l} = \hat{\mathbf{u}}^*_{N-i,j,N-k,l}$
    $\hat{\mathbf{u}}_{i,N-j,N-k,l} = \hat{\mathbf{u}}^*_{N-i,j,k,N-l}$
end for
for i,j,k,l in $\{0,N/2\}$ do
    set imaginary parts of $\hat{\mathbf{u}}_{i,j,k,l}$ to zero
end for

To compute each element $\hat{\mathbf{u}}_{a,b,c,d}$ in the first loop, three independent complex random variables $X_m = r_m e^{2\pi i\theta_m}$ $(m = 1,2,3)$ are generated with normally distributed gaussian random amplitudes r_m and with uniformly distributed random phases θ_m. The components of that element are then calculated as

$$\begin{aligned}
(\hat{u}_1)_{a,b,c,d} &= \hat{h}_{11}((i,j,k),l)X_1, \\
(\hat{u}_2)_{a,b,c,d} &= \hat{h}_{21}((i,j,k),l)X_1 + \hat{h}_{22}((i,j,k),l)X_2, \\
(\hat{u}_3)_{a,b,c,d} &= \hat{h}_{31}((i,j,k),l)X_1 + \hat{h}_{32}((i,j,k),l)X_2 + \\
&\quad\ \hat{h}_{33}((i,j,k),l)X_3.
\end{aligned}$$

The functions $\hat{h}_{mn}$ are derived from the cross-spectral density functions as shown in Appendix A (Eq. 21). The velocity field is then obtained by taking three inverse DFT's:

$$\begin{aligned}
u_1 &= \text{invFFT4D}(\hat{u}_1) \\
u_2 &= \text{invFFT4D}(\hat{u}_2) \\
u_3 &= \text{invFFT4D}(\hat{u}_3).
\end{aligned}$$

The resulting velocity field is defined on a discrete lattice and is periodic in space and time. Thus even a small lattice defines a field everywhere in space-time. The spacing of this grid determines the smallest scale of the turbulence.

4 Animation of Gaseous Phenomena

Physically a gas is composed of many particles. We could therefore animate a gas by moving its particles about the wind field, but this would require a vast set of particles. We shall instead consider the *density* $\rho(\mathbf{x},t)$ of particles at space-time point $(\mathbf{x},t)$. Assuming that the particles have no effect on the wind field, the evolution of the density distribution is given by an *advection-diffusion* (A-D) equation

[2] The choice of i,j,k here as indices should not be confused with their different use above.

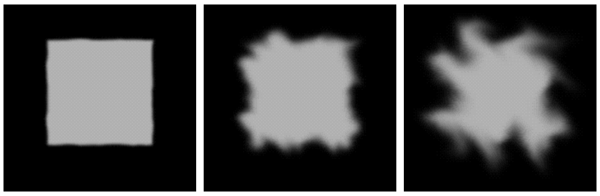

Figure 1: Evolution of a density distribution

[5] to which we have added a dissipation term:

$$\frac{\partial \rho}{\partial t} = -\mathbf{u}\nabla\rho + \kappa\nabla^2\rho - \alpha\rho. \quad (12)$$

The first term on the right hand side is the advection term that accounts for the effects of the wind field on the density. The second term accounts for molecular diffusion at rate κ. This term can also be used to model turbulent diffusion from scales smaller than the smallest scale of the modelled turbulence. The third term accounts for dissipation of density at rate α. Since the velocity $\mathbf{u}$ is given, the equation is linear in ρ and can be solved by finite differences. The density distribution is then resolved on a finite grid and can be rendered using an efficient voxel-based volume renderer [1, 6]. Figure 1 depicts the evolution of an initially square distribution evolving under the influence of a two-dimensional wind field calculated using a standard PDE solver [9]. Computations for four-dimensional wind fields become rapidly prohibitive both in computation time and memory. To obtain tractable animations we propose an alternative strategy. We shall assume that the density distribution is a weighted sum of a simple distribution f:

$$\rho(\mathbf{x}, t) = \sum_{i=1}^{n} m_i(t) f(\|\mathbf{x} - \mathbf{x}_i(t)\|, t - t_i) = \sum_{i=1}^{n} \rho_i(\mathbf{x}, t). \quad (13)$$

In other words the density distribution is a "fuzzy blobby" with time-dependent field function f, where $\mathbf{x}_i(t)$ is the centre of mass, t_i is the time at which the "blob" ρ_i is created and $m_i(t)$ is its mass. If we suppose that f is a gaussian distribution with a standard deviation σ_0 much smaller than the smallest scale of the turbulent wind field, the wind field can be assumed to be constant on each blob. The advection term therefore only moves the blob, but does not deform its shape. The movement of the blob is hence given by integrating its centre of mass over the wind field:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t_i) + \int_{t_i}^{t} \mathbf{u}(\mathbf{x}_i(s), s)\, ds, \qquad i = 1, \cdots, n. \quad (14)$$

The deformation of the shape of the blob is given by the diffusion term. Here we note that the diffusion at rate κ after time $t - t_i$ of a gaussian with variance σ_0^2 is equivalent to convolving a gaussian of variance $\kappa(t - t_i)$ with a gaussian of variance σ_0^2 (cf. [18]). Gaussians are closed under convolution, and the resulting gaussian has variance $\sigma_i^2(t) = \sigma_0^2 + \kappa(t - t_i)$:

$$f(r, t - t_i) = \frac{1}{(2\pi)^{\frac{3}{2}} \sigma_i^3(t)} \exp\left(-\frac{r^2}{2\sigma_i^2(t)}\right). \quad (15)$$

Thus f diffuses outward with variance $\sigma_i^2(t)$ that increases with t. The normalization factor $(2\pi)^{\frac{3}{2}}\sigma_i^3(t)$ guarantees that the mass of the blob is invariant under diffusion. Once the variance of a blob becomes comparable to the smallest scale of the turbulent wind field we can replace it by smaller blobs and distribute the mass equally among them. The effect of the dissipation term is an exponential decay of the masses over time:

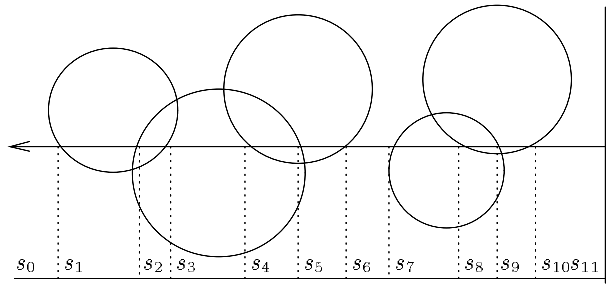

Figure 2: Subdivision of ray into intervals

$$m_i(t) = m_0 \exp\left(-\alpha(t - t_i)\right). \quad (16)$$

5 Efficient Rendering of Gas

In conventional ray-tracing, light-object interactions are only computed at object boundaries. Hence light travelling along a ray is only modified at its endpoints. In the presence of a participating medium, the light carried by a ray can be attenuated and increased: attenuation is caused by light absorbed and scattered away by the gas; an increase may arise from light scattered in the direction of the ray from other directions and by self-emission of the gas. These effects can be included into a standard ray-tracer, by modifying the intensity value returned along any ray in the ray-tree. For each such ray we first determine which blobs have domains intersecting the ray (in practice we truncate the domain of each gaussian). For each such blob we store in a sorted list the parameter value s both for the entry and exit points of the ray. This subdivides the ray into N disjoint intervals $I_i = [s_i, s_{i+1}]$ $(i = 0, \cdots, N-1)$ as illustrated in Figure 2, with $s_0 = 0$ being the origin of the ray and the s_i being points of ray/blob intersections.

Once the ordered list of blobs intersecting the ray is calculated, the intensity of light C reaching the origin of the ray is computed by shading the list from front to back [6]:

$\tau_{total} = 1$
$C = 0$
for $i = 1$ to $N - 2$ do
  $C = C + \tau_{total}(1 - \tau_i)C_i$
  $\tau_{total} = \tau_{total}\tau_i$
end for
$C = C + \tau_{total}C_N$,

Here, τ_i is the transparency of the density distribution on interval I_i, and C_i is the intensity of light emitted on that interval by the density distribution. These values are defined in Appendix B, in which we also derive the illumination model. C_N is the intensity returned by the standard ray-tracer. In case the ray is cast to determine a shadow, only τ_{total} has to be returned.

The transparency along an interval I_i due to a single blob is a function only of the distance of the ray to the centre of the blob and the endpoints s_i and s_{i+1} of the interval as shown in Figure 3. The exact relationship and an efficient way to compute them is given in Appendix B. The transparency τ_i of the interval is then computed by

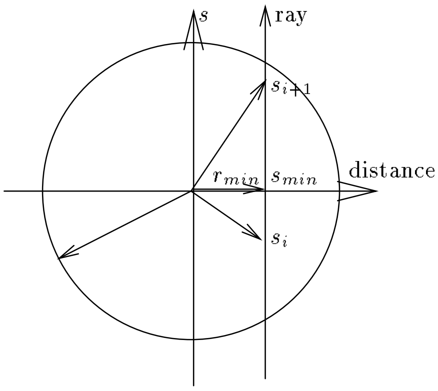

Figure 3: Calculation of transparencies τ_i

combining the transparency values calculated for each blob that intersects the ray along that interval.

Instead of testing separately for an intersection of the ray with each blob, we traverse a tree data structure of bounding spheres. The tree is constructed prior to rendering a frame as follows. First all the blobs are put in a linked list. The tree is then constructed by the following algorithm:

```
while list has at least two elements do
      for each blob b in the list do
          search for blob b' closest to b
          remove b' from list
          create new blob b'' which bounds b and b'
          set b and b' to children of b''
          replace b by b'' in list
      end for
end while
```

There are some obvious optimizations that can be made to this brute-force algorithm, such as non-binary blob groupings and the use of a k-d tree to accelerate the search, but the cost of ray tracing overwhelms even brute-force preprocessing cost. On average, the use of the tree data structure has reduced rendering times by an order of magnitude. The tree can be thought of as a multi-scale representation of the density distribution and hence could be used to render the distribution at different levels of detail.

6 Interactive Field Modelling/Results

In our implementation, modelling wind fields and their effects consists of several steps. First the energy spectrum for the spatial component of the small-scale turbulence is specified by providing numerical values for the rate ϵ and the inertial frequency k_{inertial} of the Kolmogorov energy cascade. The standard deviation σ for the temporal component of the energy spectrum is also specified. The overall energy spectrum (cf. Section 3.2) is the product of the temporal and spatial (Kolmogorov) energy spectra. A 4-D vector field is then generated (cf. Section 3.3) which can be placed in a library (although its computation is swift).

We have developed an interactive animation system in which an animator can design a complex wind field and visualize its effect on a gas density. Complex wind fields are formed by the superposition of small-scale turbulence with large-scale fields such as directional, spherical, and exponentially decaying fields. The user is also able to change the grid spacing of the small scale independently in each component of space and time, allowing the specification of non-homogeneous fields. This also permits the same prototypical small-scale field to be given different behaviours in different contexts (which is precisely what has been for the images shown below).

Our animation system also simulates the effect of a wind field on a gas. A specific gaseous phenomenon is specified as a particle system characterized by the following values: the region over which blobs of particles are born, their birth rate, and the initial standard deviation and the initial mass of each blob. During a simulation, the system introduces blobs at the given rate, animates their motion by advection, modifies the standard deviations by diffusion and the masses by dissipation, as described in Section 4. Additionally, particles can be given illumination parameters such as a colour. In this modelling step the centre of each blob is depicted (with intensity modulated by parameters such as duration), but positions and other data can be piped into a high-quality renderer for image synthesis. About $6,000$ particles can be animated in real time on an SGI Indigo.

The parameters needed for rendering include (Appendix B): the extinction coefficient κ_t, which describes the decay of light in inverse proportion to distance; the albedo $\Omega \in [0,1]$, which defines the proportion of light scattered at a given point; the phase function p, giving the spherical distribution of scattered light; and self-emission Q, which is the amount of light emitted by a blob at a given position. The illumination computation for gas densities at a resolution of 640×480 typically requires from one to ten minutes, although 1-2 hour computations are possible when rendering scenes of high optical complexity.

For the images presented below, we have assumed that the phase function is constant and we have ignored shadows cast onto the density distribution for all but one image sequence. In all simulations the same statistical parameters were used for the small scale component: $\epsilon = 1$, $k_{\mathrm{inertial}} = 4$ and $\sigma = 1$.

Steam from a mug: One global directional wind field was used to model the rising of the steam due to thermals. The particles were generated uniformly on a disk.

Psychedelic steam: Three trails of smoke of different colours were combined. As for the steam we used a directional wind field, this time tilted in the direction of the teapot spout. Particles were again generated on small disks.

Cigarette smoke: Two smoke trails originating from the tip of a cigarette are derived from the similar small-scale turbulence as the steam with a directional heat source.

Interaction of a sphere with smoke: This simulation shows how objects can interact with our wind field model. Instead of testing for collision of particles with the objects, we define a repulsion field around each object. We modelled the repulsion force by a radial potential field. The sphere is moved along a path given by a spline curve. Note that this image sequence depicts self-shadowing.

Three-dimensional morphing: The cylindrical range data of two human heads was converted into two sets of blobs and input to the animation system. The scene was illuminated by setting the self-illumination parameter (Q in Eq. 24) of each blob to the illumination given by the range data. The albedo was set to zero and dissipation was set to a large value to allow rapid dissolution of each set of blobs (with one run in reverse).

7 Conclusions and Extensions

We have presented a new model for the visual simulation of gaseous phenomena in turbulent wind fields. Our model provides an animator with control over both the large-scale motion and the statistical features of the small-scale turbulence. This model has been successfully applied to the animation of gaseous phenomena. Our model, however, can be applied to many other phenomena resulting from the interactions of objects with a wind field. For example, the wind field model can be included in any existing physically-based animation system. Our model can in fact generate a random vector field of any dimension, not only three-dimensional vector fields with a four dimensional domain. The derivation of the algorithm can be adapted in a straightforward manner. Our fast rendering algorithm can be used to visualize sparsely sampled data. The rendering of the heads in the morphing animation is a good example. Also our animation system could be used to visualize wind fields calculated by direct numerical simulation for fluid dynamics applications.

There are many other extensions to our model that we will explore in future research. We have assumed that the large scale motions of the wind do not modify the small turbulent scale. This is implausible. One possible solution is to warp the domain of the turbulent scale according to the large scales. We would require the use of a global deformation algorithm. Also it is possible to use a physical model for the large scales. A numerical technique in computational fluid dynamics known as *Large Eddie Simulation (LES)* solves the Navier-Stokes equations on a coarse grid using a statistical model for the small scales [11]. However, a physical simulation might not be relevant in computer graphics when a specific behaviour is intended.

A Inverse FFT Method Derivation

A white noise velocity field has cross-spectral density functions defined by:[3]

$$\Phi^w_{kl}(\mathbf{k},\omega) = \langle \hat{w}^*_k(\mathbf{k},\omega)\hat{w}_l(\mathbf{k},\omega)\rangle = \delta_{kl}. \quad (17)$$

A random field with cross-spectral density functions Φ_{ij} can be obtained by *cross-convolving* this white noise with a set of deterministic kernels h_{kl}:

$$u_k(\mathbf{x},t) = \sum_{l=1}^{3} \int_{\mathbf{R}^3}\int_{-\infty}^{\infty} h_{kl}(\mathbf{x}-\mathbf{y}, t-s) w_l(\mathbf{y},s)\, ds\, d\mathbf{y}, \quad (18)$$

which in the Fourier domain becomes

$$\hat{u}_k(\mathbf{k},\omega) = \sum_{l=1}^{3} \hat{h}_{kl}(\mathbf{k},\omega)\hat{w}_l(\mathbf{k},\omega). \quad (19)$$

We obtain an equation for the transformed kernels $\hat{h}_{kl}$ in terms of the cross-spectral density functions Φ_{ij} by inserting the expressions for the Fourier velocity components $\hat{u}_i$ and $\hat{u}_j$ given by Eq. 19 into the definition of the cross-spectral density function Φ_{ij} (see Eq. 6).

$$\begin{aligned}\Phi_{ij}(\mathbf{k},\omega) &= \langle \hat{u}^*_i(\mathbf{k},\omega)\hat{u}_j(\mathbf{k},\omega)\rangle \\ &= \sum_{k=1}^{3}\sum_{l=1}^{3} \hat{h}^*_{ik}(\mathbf{k},\omega)\hat{h}_{jk}(\mathbf{k},\omega)\Phi^w_{kl}(\mathbf{k},\omega) \\ &= \sum_{n=1}^{3} \hat{h}^*_{in}(\mathbf{k},\omega)\hat{h}_{jn}(\mathbf{k},\omega). \quad (20)\end{aligned}$$

We thus have 9 equations for the 9 kernels $\hat{h}_{kl}$ in terms of the cross-spectral density functions. Because of the symmetry of the cross-spectral density functions ($\Phi_{ij} = \Phi_{ji}$), only 6 of these kernels are independent and three kernels can be chosen arbitrarily. If we set $\hat{h}_{12} = \hat{h}_{13} = \hat{h}_{23} = 0$, then the system of equations given by Eq. 20 becomes diagonal and can easily be solved as follows.

$$\begin{aligned}\hat{h}_{11} &= \sqrt{\Phi_{11}}, \quad \hat{h}_{21} = \frac{\Phi_{21}}{\hat{h}_{11}}, \quad \hat{h}_{31} = \frac{\Phi_{31}}{\hat{h}_{11}} \\ \hat{h}_{22} &= \sqrt{\Phi_{22} - \hat{h}^2_{21}}, \quad \hat{h}_{32} = \frac{\Phi_{32} - \hat{h}_{31}\hat{h}_{21}}{\hat{h}_{22}} \\ \hat{h}_{33} &= \sqrt{\Phi_{33} - \hat{h}^2_{31} - \hat{h}^2_{32}}. \quad (21)\end{aligned}$$

B Illumination Model

Consider a ray $\mathbf{x}_s = O + sD$, with origin O and direction D. Let C_N be the intensity of light reaching O along the ray from point $\mathbf{x}_b$ in the absence of a density distribution (i.e., given by a conventional ray-tracer). If we ignore multiple scattering effects, then the illumination C_0 reaching point O along the ray for each visible wavelength λ is [3]

$$C^\lambda_0 = \int_0^b \tau^\lambda(0,s)\rho(\mathbf{x}_s)\kappa^\lambda_t C^\lambda(\mathbf{x}_s)\, ds, \quad (22)$$

where

$$\tau^\lambda(s',s'') = \exp\left(-\kappa^\lambda_t \int_{s'}^{s''} \rho(\mathbf{x}_s)\, ds\right), \quad (23)$$

$$C^\lambda(\mathbf{x}_s) = \Omega^\lambda L^\lambda(\mathbf{x}_s) + (1-\Omega^\lambda)Q^\lambda(\mathbf{x}_s), \quad (24)$$

and κ_t is the *extinction coefficient*, and Ω is the *albedo*. The term $L(\mathbf{x}_s)$ is the contribution due to N_l light sources:

$$L^\lambda(\mathbf{x}_s) = \sum_{k=1}^{N_l} p^\lambda(cos\theta_k(\mathbf{x}_s))S_k(\mathbf{x}_s)L^\lambda_k, \quad (25)$$

where p is the phase function characterizing the scattering properties of the density distribution, the θ_k are the angles between the ray and the vectors pointing to the light sources, S_k determines if the light source is in shadow and L_k is the colour of the light source. The term $Q^\lambda(\mathbf{x}_s)$ accounts for self-emission and can be used to approximate the effects of multiple scattering. If we assume that $C^\lambda(\mathbf{x}_s) = C^\lambda_i$ is constant on each interval I_i, which is reasonable in the case of many small blobs, then Eq. 22 becomes

$$\begin{aligned}C^\lambda_0 &= \sum_{i=0}^{N-1} C^\lambda_i \int_{s_i}^{s_{i+1}} \tau^\lambda(0,s)\rho(\mathbf{x}_s)\kappa^\lambda_t\, ds \\ &= \sum_{i=0}^{N-1} C^\lambda_i \left(\tau^\lambda(0,s_i) - \tau^\lambda(0,s_{i+1})\right). \quad (26)\end{aligned}$$

If we define $\tau^\lambda_i = \tau^\lambda(s_i, s_{i+1})$ as the transparency along interval I_i then the equation becomes

$$C^\lambda_0 = \sum_{i=0}^{N-1}\left(\prod_{j=0}^{i-1}\tau^\lambda_j\right) C^\lambda_i \left(1-\tau^\lambda_i\right). \quad (27)$$

We now show how the integral occurring in the calculations of the transparencies τ^λ_i can be computed efficiently. Let us assume that the blobs $\rho_{j_1}, \cdots, \rho_{j_{n_i}}$ intersect the ray on interval I_i. The transparency on interval I_i is then

$$\tau^\lambda_i = \exp\left(-\kappa^\lambda_t \sum_{k=1}^{n_i} \int_{s_i}^{s_{i+1}} \rho_{j_k}(\mathbf{x}_s)\, ds\right). \quad (28)$$

[3] All subscripted indices in this appendix take on the values 1, 2, 3.

As we render for a particular frame in time we define $\sigma_j^2 = \sigma_0^2 + \kappa(t - t_j)$ and $m_j = m_j(t)$. Using these definitions, each integral in Eq. 28 can be written as [8]:

$$\int_{s_i}^{s_{i+1}} \rho_j(\mathbf{x}_s)\, ds = \frac{m_j}{(2\pi)^{\frac{3}{2}}\sigma_j^3} \int_{s_i}^{s_{i+1}} \exp\left(-\frac{r_{min}^2 + (s - s_{min})^2}{2\sigma_j^2}\right)\, ds$$

$$= \frac{m_j}{(2\pi)^{\frac{3}{2}}\sigma_j^2} \exp\left(-\frac{r_{min}^2}{2\sigma_j^2}\right)\left(T\left(\frac{s_{i+1} - s_{min}}{\sigma_j}\right) - T\left(\frac{s_i - s_{min}}{\sigma_j}\right)\right).$$

The first equality results from the geometry of Figure 3. The function T is the following integral:

$$T(s) = \int_0^s \exp\left(-\frac{u^2}{2}\right)\, du, \qquad (29)$$

and can be precomputed and stored in a table for efficiency.

References

[1] D. S. Ebert and R. E. Parent. "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):357–366, August 1990.

[2] A. F. Fournier, D. Fussell, and L. Carpenter. "Computer Rendering of Stochastic Models". *Communications of the ACM*, 25(6):371–384, June 1982.

[3] A. Ishimaru. *VOLUME 1. Wave Propagation and Scattering in Random Media. Single Scattering and Transport Theory.* Academic Press, New York, 1978.

[4] M. Kass and G. Miller. "Rapid, Stable Fluid Dynamics for Computer Graphics". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):49–57, August 1990.

[5] M. Lesieur. *Turbulence in Fluids: Stochastic and Numerical Modelling.* Kluwer Academic Publisher, Dordrecht, The Netherlands, 1990.

[6] M. Levoy. "Efficient Ray Tracing of Volume Data". *ACM Transactions on Computer Graphics*, 9(3):245–261, July 1990.

[7] J. P. Lewis. "Generalized Stochastic Subdivision". *ACM Transaction on Graphics*, 6(3):167–190, July 1987.

[8] N. Max, R. Crawfis, and D. Williams. "Visualizing Wind Velocities by Advecting Cloud Textures". In *Proceedings of Visualization '92*, pages 179–183, Los Alamitos CA, October 1992. IEEE CS Press.

[9] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988.

[10] W. T. Reeves and R. Blau. "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):313–322, July 1985.

[11] R. S. Rogallo and P. Moin. "Numerical Simulation of Turbulent Flows". *Annual Review of Fluid Mechanics*, 16:99–137, 1984.

[12] G. Sakas. "Modeling and Animating Turbulent Gaseous Phenomena Using Spectral Synthesis". *The Visual Computer*, 9:200–212, 1993.

[13] M. Shinya and A. Fournier. "Stochastic Motion - Motion Under the Influence of Wind". In *Proceedings of Eurographics '92*, pages 119–128, September 1992.

[14] K. Sims. "Particle Animation and Rendering Using Data Parallel Computation". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):405–413, August 1990.

[15] E. Vanmarcke. *Random Fields.* MIT Press, Cambridge, Massachussetts, 1983.

[16] R. P. Voss. "Fractal Forgeries". In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics.* Springer-Verlag, 1985.

[17] J. Wejchert and D. Haumann. "Animation Aerodynamics". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):19–22, July 1991.

[18] A. Witkin and M. Kass. "Reaction-Diffusion Textures". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):299–308, July 1991.

Sphere interacting with a gas (note the shadowing)

A strange brew

The lonely cigarette

From David to Heidi