Contents

List of Figures

Index of Notation

α	absorption coefficient	21
B_λ	black body radiation	71
B_i	blob density of mass	58
$\mathcal{B}$	σ-algebra	28
$\mathcal{B}_{\mathbf{f}}$	σ-algebra induced by a random variable $\mathbf{f}$	29
c	speed of light	16
c_k	point closest on the ray to the center of a blob	79
C_p	specific heat of a fluid or gas	24
$\mathbf{C}_{\mathbf{f}}$	covariance of a random variable or function $\mathbf{f}$	31
d	dimension of the domain of a function	22
d_k	distance of ray to the center of a blob	79
D_f	fractal dimension of a random fractal	37
$\mathbf{D}_{\mathbf{f}}$	structure function of a random function $\mathbf{f}$	35
δ	delta Dirac function	37
Δ_k	overlap of truncated blob and the ray	79
E	isotropic energy spectrum of a random function	39
E_0	energy spectrum of a random function	36
E_a	activation energy	124
E_i	kinetic energy of a particle	15
E_λ	characterizes the colour emission of a material	71
$\mathcal{E}$	event space	28
ϵ	rate of dissipation of kinetic energy	50
	magnitude of perturbation	100
η	clamping value	100
$\mathbf{f}$	random variable	29
$\bar{\mathbf{f}}(\mathbf{x})$	mean of a random function	34
$\mathbf{f}_\omega$	realization of a random variable	29
$\mathbf{f}_\sigma$	smoothed version of a function	56
F_{12}	form factor between two elements (patch or blob)	86
$F_{\mathbf{f}}$	probability distribution function of the random variable $\mathbf{f}$	29
$\mathbf{F}_i$	total force exerted on a particle	14
$\mathbf{F}_i^{ext}$	external force exerted on a particle	14
$\mathbf{F}_i^{bnd}$	force exerted on a particle due to boundary surfaces	14
$\mathbf{F}_{ij}$	interparticle force	14
F_N	N-point probability distribution of a random function	34
F_η	clamping function	100
φ	probability density of a particle system	16

φ_1	one-particle distribution of a particle system	16
φ_2	two-particle distribution of a particle system	17
φ_{brdf}	bidirectional reflectance density function	72
$\varphi_{\mathbf{f}}$	probability density function of a random variable $\mathbf{f}$	30
ϕ	one-speed one-particle distribution of a particle system	18
ϕ_0	one-speed one-particle distribution of a particle system at time $t = 0$	18
ϕ_{in}	incident particle probability density	19
ϕ_{out}	scattered particle probability density	19
$\mathbf{\Phi}$	interparticle potential	14
$\mathbf{\Phi}_0$	depth of the Lennard-Jones interparticle potential	14
Φ_{LJ}	Lennard-Jones interparticle potential	14
Φ_m	isotropic component of the spectral density of an isotropic random function	40
$\mathbf{g}$	difference between two velocities	20
G_σ	Gaussian function with standard deviation σ and zero mean	51
Γ	gamma function	39
$\Gamma(a,b)$	optical depth of blobs along a ray	78
$\Gamma_k(a,b)$	optical depth of one blob along a ray	78
h	Planck's constant	16
	grid spacing	22
$\mathbf{H}$	filter kernel	42
I^0	average intensity over all angles	76
I_d	diffuse intensity	72
I_{diff}	diffuse component of angular dependence of intensity	76
I_{in}	intensity incident on a surface	72
I_{out}	intensity reflected from a surface	72
I_{ri}	reduced incident intensity	72
I_{spec}	specular component of angular dependence of intensity	76
I_ν	radiant intensity of light	16
$\mathbf{I}^1$	average intensity flux over all angles	76
J	source intensity	74
J_d	diffuse source intensity	74
J_{ri}	source term due to the reduced incident intensity	73
$\bar{J}$	average source intensity	74
$\mathbf{J}$	directional average of the one-particle distribution flux	21
k	temperature conductance of a fluid or gas	24
	length of a wavenumber $\mathbf{k}$	39
	Stefan-Boltzmann's constant	71
K_a	absorption coefficient in transport equation	18
K_s	scattering coefficient in transport equation	18
K_t	extinction coefficient in transport equation	18
$\mathbf{k}$	wave number	36
κ	diffusion coefficient	21
κ_T	thermal diffusivity	124
l_0	smallest eddie size	50
L	largest eddie size	50
L_T	loss of temperature	124

$\mathcal{L}$	linear operator of diffusion	22
λ	wavelength of the intensity of light	70
m	mass of each particle	14
m_i	mass of a blob	58
$\mathbf{M}_h$	discretization of the diffusion operator $\mathcal{L}$	22
μ	cosine between two unit vectors	71
μ_2	indicates either total forward or total backward scattering	77
$\bar{\mu}$	first moment of the phase function	71
N	number of particles	14
	number of samples	22
	number of basis functions	23
NS	denotes the Navier-Stokes equations	49
$\mathbf{n}$	normal to a surface	72
ν	frequency of light	16
	coefficient of viscosity of a fluid	24
ν_a	combustion "frequency"	124
ν_b	smoke creation "frequency"	126
ω	element of an event space	29
Ω_λ	albedo	71
p	pressure field of a fluid or a gas	24
p_λ	phase function	71
P	probability measure	28
$P_\mathbf{f}$	probability measure induced by the random variable $\mathbf{f}$	29
$\mathbf{P}$	projector operator	40
Q	particle emission	18
r	length of the spacing $\mathbf{r}$	38
r_0	radius beyond which the Lennard-Jones potential becomes attractive	14
R	universal gas constant	124
$\mathbf{r}$	spacing between two points	36
ρ	density of mass	16
$\mathbf{s}$	direction vector	18
S	source term in a diffusion equation	21
$\mathbf{S_f}$	spectral density function of a random function $\mathbf{f}$	36
$\mathcal{S}$	scattering functional	71
σ	particle collision kernel	20
σ_a	absorption cross-section	71
σ_s	scattering cross-section	71
σ_t	extinction cross-section	71
$\sigma_t r$	transport cross-section	92
$\bar{\sigma}_a$	mean absorption cross-section	125
Σ	scattering kernel giving the distribution of scattered particles	18
Σ_S	reflection probability distribution of surface S	19
t	time variable	14
T	temperature field of a fluid or gas	24
	kinetic energy transfer	50
T_a	activation temperature	124

T_b	smoke creation temperature	126
τ	transparency	73
$\mathbf{u}$	velocity field of a fluid or gas	24
$u^{\pm}$	limits of the clamping when stochastic rendering	102
U	directional average of the one-particle distribution	21
v_i	magnitude of the velocity of a particle	15
$\mathbf{v}_i$	velocity of a particle	14
$\mathbf{x}$	spatial position	16
$\mathbf{x}_i$	position of a particle	14
$\mathbf{x}_u$	ray in space indexed by u	72
$\mathbf{x}^{-1}$	backtraced point through the motion field	65
$\mathbf{X}_N$	state of a system of N particles	14
W	smoothing kernel	56
$\mathbf{w}_\omega$	realization of a white noise random function	44

Chapter 1

Introduction

The eye which is called the window of the soul, is the chief means whereby the understanding may most fully and abundantly appreciate the infinite works of nature. *Leonardo Da Vinci*

We see nothing truly until we understand it. *John Constable*

1.1 Motivation

The appearance of natural phenomena has always intrigued humankind. This fascination is evident from the vast array of depictions of natural phenomena created, ranging from early cave paintings to impressionistic masterworks. With the advent of photography and film, many aspects of natural phenomena can be depicted by a semi-automatic procedure. There are, however, many reasons why such depictions are not always satisfying. Movies require special effects not commonly found in nature or which occur only very rarely. Also, even if the real phenomenon can be found, controlling it can be a difficult and even dangerous task. It is necessary, then, to artificially create phenomena such as explosions for entertainment. In response to the need to evoke and control nature, scientists have proposed many descriptive physical models of natural phenomena. Although scientists share the same fascination of nature as artists, scientists need to command or predict the behaviour of certain phenomena. Here, physical models can be used to simulate the phenomenon on a computer. In fact, a *visual simulation* of the phenomenon is obtained when the data of a simulation is mapped into a visual depiction. The simulation itself, however, is not necessarily based on a physical model. In the context of this thesis, both a cartoon and a visualization of a physical simulation are referred to as visual simulations.

The visual simulation of virtual environments has many practical applications. In addition to art and entertainment these include flight simulation and scientific visualization. Natural phenomena such as smoke, clouds, grass and hair, are ubiquitous. It is therefore important to design convincing visual simulations for them. For example, in flight simulators, it is critical that clouds and terrain be carefully depicted because they serve as vital visual cues to a pilot. However, this is not the only constraint imposed on a simulation. In the entertainment industry, it is fundamental to be able to choreograph certain phenomena in order to create a desired mood. The simulation of natural phenomena in the context of computer graphics is therefore subjected to two constraints: visual consistency and user control. Finding an optimal compromise between these two constraints for some classes of natural phenomena is the goal of this thesis.

1.2 Physical Simulations

The nearest a scientist can get to an apple is to measure its weight, size, shape, location, taste. The nearest a percept can get to the stimulus "apple" is to represent it through a specific pattern of such general sensory qualities as roundness, heaviness, fruity taste, greenness. *Rudolf Arnheim*

The science of anatomy, the sciences of projective geometry and of optics were called in to hasten the experimentation towards recognizable images. In the end, as we know, science overtook art in this respect through the evolution of photography, the colour film and the wide screen. *E. H. Gombrich*

The appearance and evolution of natural phenomena have been studied extensively in the physical sciences. Physical properties of the phenomenon are quantified and mathematical equations are derived to describe their evolution. A particular method of solution of these equations defines a physical simulation. Typically such a simulation then is run from an initial state and subjected to boundary conditions. The effectiveness of such a simulation is measured by comparing its results to empirical data. Physical simulations alone are usually poor visual simulations for the following reasons:

- The behaviour of the simulation can only be controlled by the initial state and the boundary conditions. The granularity of the behaviour imposed by these conditions is very large. For example, how can we solve the initial angular velocity of the roll of a die such that it lands on a given face?
- Practical physical simulations do not exist for most turbulent phenomena. This is due both to the lack of models for turbulence and to current computational limitations.
- Physical simulations usually provide redundant data which have no impact on its visual appearance. The prevailing problem in this case is to determine what portion of the physical simulation (or what transformation) is sufficient for a visual simulation.
- It is difficult to validate the results of a physical simulation in the context of virtual environments. Typically, experimental data is deficient or is irrelevant to the goal of the simulation. Therefore, use of a physical model is often justified only a priori.

Despite these drawbacks, physical models are crucial to a methodology for modelling natural phenomena. It is arduous to achieve a convincing motion or behaviour by key-framing the simulation, where the animator specifies each step of the simulation exactly. On the other hand, complex motions but poor control can be achieved (assuming enough computing power is available) by a physical simulation. For example, turbulent trails of smoke are often evoked in traditional animation by a single brush-stroke. A close examination of a trail of smoke will demonstrate many quite intricate patterns evolving in a turbulent fashion. Good visual simulations should achieve these effects through an automatic process which does not require a high degree of expertise of the user. Therefore, visual simulations usually are a trade off between control and complexity, and it is to be expected that such simulations borrow techniques from both traditional non-physical techniques and physical simulations.

1.3 Realism and Accuracy

There are different solutions to the problem of representing three-dimensional objects in a two-dimensional plane. Each method has its virtues and its drawbacks, and which is preferable depends on the visual and philosophical requirements of a particular time and place. *Rudolf Arnheim*

In a word, what scientific activity achieves is not real, but ideal. *Edmund Husserl*

As suggested, the "success" of a visual simulation depends on many factors. One factor which is often singled out in computer graphics is "realism". Indeed, the "realism" of the animation submitted with a paper to a computer graphics journal or a conference is often an important factor in its acceptance or rejection. But, the notion of realism is highly subjective and varies wildly among individuals. Nevertheless, several possible solutions have been proposed to this problem. One solution is to set up a Turing test-like experiment, in which a sample of observers compare a visual simulation to a "real phenomenon" [57]. Because visual simulations are often viewed on colour monitors, it has been suggested that the "real phenomenon" should actually be a photograph or a film of a phenomenon [18]. Some researchers already have provided such comparisons with their results [65]. The experiment then measures how effective the visual simulation is at mimicking a depiction of the phenomenon through video or film. In fact, the visual simulation should include an accurate model of the camera. In case the results of such an experiment show that there is no statistical difference, then we can conclude that the visual simulation is indeed "realistic". In practice, however, these conditions are too stringent and not always feasible. Indeed, in many situations, a visual simulation might appear realistic to a large audience only when seen in isolation, and not next to the filmed version. As well, in some instances it is quite impossible to obtain footage of the genuine phenomenon. Real action footage of dinosaurs would have been impossible to obtain for Jurassic Park, for example. However, the film was able to convince most of its audience that the dinosaurs were authentic. Additionally, having phenomena which behave unrealistically is sometimes a desired feature of an animation. In this situation, there is no possibility of finding an objective experiment to measure the success of the simulation.

Another solution to the problem of determining the "realism" of a visual simulation has been proposed mainly in the global illumination literature. It states that the exact solution to a physical equation for the propagation of light for a particular scene is considered to be the "real phenomenon". This precise solution can either be computed for a simple case using an analytical solution or it can be computed by a method which is known to be accurate. The success of new methods is then measured through their efficiency and the proximity of their results to the exact solution, and hence to the "real phenomenon". Although this method has the advantage of allowing objective evaluation of submitted results, it has several drawbacks. First, all physical equations are merely approximations of the phenomenon of the propagation of light. The same can be said for all physical models. For instance, diffraction effects cannot be modelled using the radiosity equations. Second, the accuracy is measured by comparing absolute values of the intensity of light. It is well known that our visual system mainly perceives differences in intensities rather than absolute values. To illustrate, multiplying all the intensities obtained from a simulation by a constant value would, after visual accommodation, look similar to an observer but would be inaccurate as compared to the exact solution. Indeed, the accuracy of a visual simulation is usually hard to model as it depends on many factors, including the state of mind of the observer. Instead of numerical accuracy as compared with a precise solution, it is often sufficient that the visual data generated by the simulation be *consistent*. For instance, when calculating the intensity of light caused by a moving light, it is more important that the distribution of light does not vary wildly from frame to frame and that the simulation does not produce visible artifacts. In the latter example, a solution which is efficient and consistent but does not solve the transport equation accurately is usually sufficient in practical applications. These methods are usually referred to as "hacks", but perceptually they may be as accurate as the physically based methods.

Regardless of the "realism" of the results, an important aspect of a visual simulation is the underlying model. A model which is only capable of producing convincing depictions of a very restricted class of phenomena might be of little interest to computer graphics. Conversely, a simple model which can be applied to a wide range of phenomena but has not yet produced convincing results might be preferable. The model need not produce "realistic" results only. A good example of the latter is the genetic texture algorithm of Sims [94].

We will not pursue these issues further in this thesis. The "success" of our visual simulations depends on the tractability of our algorithms: real-time response time during the design process and "reasonable" computation time during the final rendering of the animation.[1] We do not claim that the results are "realistic" in the sense that they are undistinguishable from a photograph or film of the real phenomenon. However, the results obtained by the method have received acceptance in the computer graphics community and beyond through publication and technical slides. Furthermore, the algorithms have been implemented in a widely used commercial computer graphics package.

1.4 Methodology

The success of any physical investigation depends upon the judicious selection of what is to be observed as of primary importance. *James Maxwell*

An important visual fact of most natural phenomena is that they appear and behave differently at various scales. A river can be described globally by a curve which follows the centre of the flow when seen from far away. A closer look at the edge of the river, however, reveals a complex flow which is hard to describe. An animator can easily model the larger scales but cannot model the turbulent scales easily. The turbulence is often evoked using simple metaphors such as swirling lines. From the point of view of control, this separation is therefore natural. The physical description of a phenomenon can vary from scale to scale. At the microscopic level, a river can be modelled by the interaction of water molecules. This description is rarely used in practice since the visible structures of water are several orders of magnitudes larger than the size of a molecule. Liquids are then modelled by density, velocity and temperature fields. In theory, the equations governing the evolution of these fields can be solved for all scales of the flow. However, for complicated flows such as water, this is computationally infeasible. For practical reasons, the smaller scales are modelled by

[1]What is considered unreasonable today may be considered reasonable tomorrow. In our case it means being able to generate a 10 second animation within at least a couple of days using the 5 Irises available in the lab.

a stochastic model. The idea of separating a phenomenon into different scales of detail is therefore important both in controlling and in modelling the phenomenon. Multiple scales permit the separation of control from the physical simulation. In the river example, a user can sketch the centre flow of the river and provide turbulence qualitatively. From this information, a physical simulation can be driven. Moreover, the decomposition of a phenomenon into different scales can be used to make certain algorithms more efficient. For instance, smaller scales can be discarded when a phenomenon is viewed from far away.

A natural mathematical framework for the modelling of complex natural phenomena is the theory of random functions. In this theory a phenomenon is modelled as a function whose exact form and evolution is unknown. The function is described by macroscopic parameters which model aggregate properties of the phenomenon. This framework is both important in the design and the physical simulation of natural phenomena. In the former, an animator controls macroscopic parameters which can be mapped onto comprehensive design concepts. For example, the complicated structure of terrain can be characterized by a single “roughness” parameter. The exact detail is then generated through a stochastic terrain synthesis procedure. In physical simulations, simple equations for the evolution of the average values of the phenomenon can be developed. The turbulent smaller scales can then be generated by a random function, alleviating the need of computationally expensive direct simulations of these scales.

1.5 Contributions

A distant tree is not a flat and even piece of colour, but a more or less globular mass of a downy or bloomy texture, partly passing into a misty vagueness. I find, practically, this lovely softness of far-away trees the most difficult of all characters to reach... *John Ruskin*

1.5.1 Insights

The basic approach outlined in the previous section has been successfully applied to the visual simulation of a broad range of natural phenomena. Specifically, the idea of separating a phenomenon into a smooth component and a small scale turbulent component has proven to be effective in both animation and rendering. It was found that for many phenomena, user control could be easily achieved through motion fields. A user specifies entirely the large scale of fields using a set of simple fields. The use of such fields include directional fields, point fields and vortex fields. These fields can be effective in modelling a particular effect. To achieve complex motions, the animator has control over a random motion field whose parameters are derived from a physical model of turbulence. Once a particular field has been designed, the evolution of various physical quantities subjected to this field can be simulated. In most cases the physical model for the large scales can be described by an advection-diffusion type equation for which efficient simulations can be devised. In particular, by using the paradigms of particle systems, such an evolution can be displayed in real time on workstations, which is crucial in the design process of a particular motion. Examples of physical quantities which were simulated in this manner include: densities of gas particles, temperature fields of flames, the intensity field of gaseous phenomena and hair filaments. An efficient approximate solution of the advection-diffusion equation is achieved by representing the fields on an unordered set of points, rather than on a regular grid.

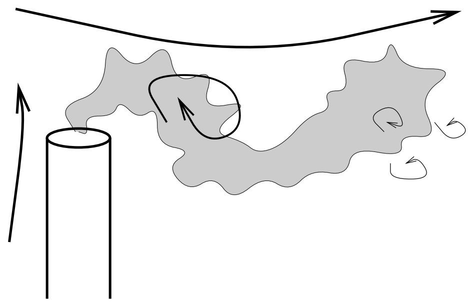

Figure 1.1: Sketch of smoke coming out of a stack. The arrows indicate the different type of motion fields.

The synthesis of turbulent motion fields is one example of a more general procedure to simulate certain phenomena directly from a stochastic model. Once quantities such as the average value and the correlation function are known, a stochastic synthesis algorithm can generate the phenomenon directly without the need of a physical simulation. An important area of research is then to determine stochastic models for various phenomena. In many cases, these models are related to other models through a transformation process. For instance, a stochastic model for the motion of a pendulum subjected to a random forcing term can be related to the statistics of the forcing term. These ideas were used to derive a stochastic model describing the intensity field resulting from a random density distribution. In this case there is a simple relationship between the mean value and correlation of the density field and the corresponding mean value and correlation of the intensity field. The advantage of this approach is that a realization of the intensity field is generated directly, without having to calculate the propagation of light through the density field directly.

In general, the use of stochastic models for natural phenomena can improve efficiency at many stages of a simulation. The parameters of a stochastic model can be mapped onto meaningful design paradigms. The physical simulation need only determine average quantities of the model. The smaller scales, which are computationally expensive to compute, are generated directly using a stochastic synthesis technique. Finally, the rendering of a specific phenomenon can be sped up by using a stochastic model for the propagation of light.

1.5.2 An Example

In order to make the methodology outlined more concrete, we give an example of a phenomenon which is modelled using these techniques. Assume an animator desires to depict billowy smoke coming out of a smoke stack. The sketch depicted in Figure 1.1 is a very coarse representation of what an animator could have in mind. The global motion of the smoke is outlined by the thick arrows. The stack is placed on the left and the smoke travels to the right through the effect of a directional wind field. In addition, the smoke rises due to the heat coming from the stack. The turbulent motion of the smoke is evoked in the sketch by small swirls indicating the scale of the perturbation. In the sketch we can identify two scales of turbulence. One scale corresponds to the wavy motion of the trail of smoke, the second, smaller scale, corresponds to the texture of the smoke. The thickness and transparency of

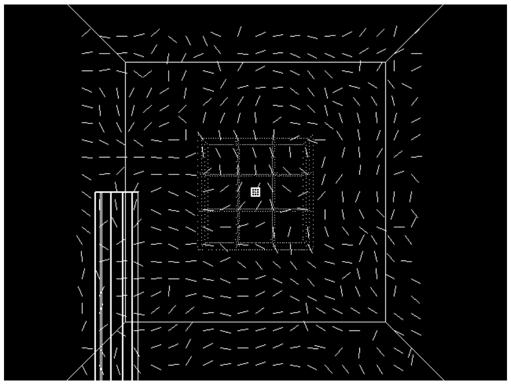

Figure 1.2: Intercative modelling of the motion fields.

the smoke at this stage is indicated by a simple outline and a single grey-level.

Once the animator has a clear picture of the phenomenon in mind he/she can start to model it using our techniques. Using smooth motion fields the animator specifies both the directional wind field and the heat field. The perturbation is added by using two precomputed turbulent motion fields stored in grids. The animator obtains the desired scale of the turbulence by adjusting the spacing of these grids. Figure 1.2 shows an interactive display of the wind fields. Notice the two grided cubes which depict the scale of the turbulence. Once the wind fields are specified, the animator can emit smoke at the tip of the stack. The evolution of the smoke is then entirely determined by the motion fields and the physical properties of the gas. The animator can then adjusts these parameters, such as the rate of diffusion, the rate of dissipation and the initial density. Figure 1.3 depicts the smoke with different settings of the parameters. In particular, the appearance of the smoke is coarsely approximated by a superposition of blobs. When the animator is content with the general "look" of the simulation, high quality renderings can be computed for each frame. The illumination is entirely calculated by a global illumination rendering algorithm which takes into account the light sources, emitting/reflecting nearby surfaces and volumetric light effects. Renderings of various degrees of complexity can be achieved by enabling or disabling certain interactions. For example, Figure 1.4 shows four different renderings of the same smoke. The top/left picture shows the smoke with a constant illumination, the top/right picture includes self-shadowing caused by absorption and by out scatter by the gas. The picture on the bottom left includes the effects of multiple scattering within the smoke. The final picture includes the effects of the small scale turbulence and is obtained by warping the blobs. In general, the choice of the level of rendering is a tradeoff between computational speed and visual accuracy.

1.5.3 Results

The main results of this thesis include [2]:

- Algorithms to generate realizations of random functions with arbitrary dimensions both for its domain and values (3.3). In particular, algorithms are given to generate three-dimensional motion fields evolving over space and time (3.4.3).

[2]The section in which the result is described is given between parentheses

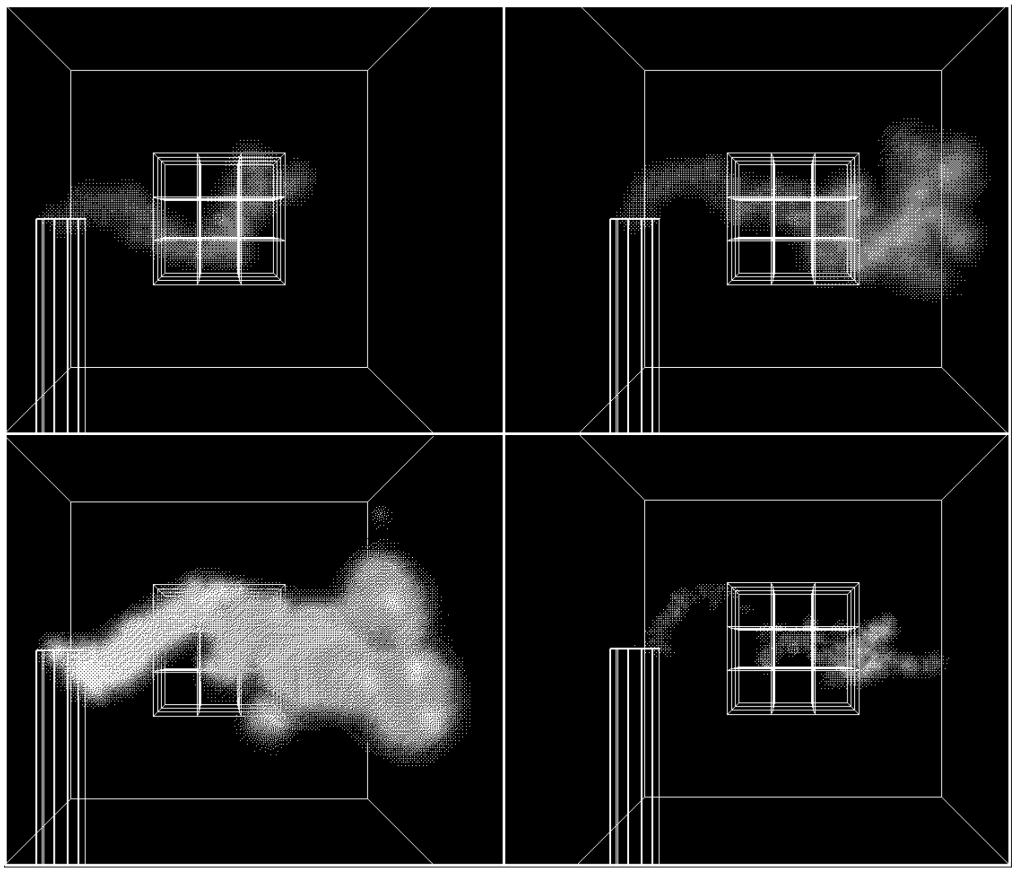

Figure 1.3: Interactive modelling of the appearance of the smoke.

- A diffusion model from transport theory to model the effects of multiple scattering in density fields (5.5.2).
- A new rendering paradigm that we call *stochastic rendering* (5.6). This method generates realizations of the intensity field directly from a stochastic model.
- A general multi-scale representation of physical quantities on a set of arbitrarily located points (4.2). This representation is used to solve steady-state diffusion equations (4.3.1) and advection-diffusion equations (4.3.2). These techniques are used to solve for the effects of multiple-scattering modelled as a diffusion process (5.5.2) and to solve for the evolution of gases, fire and hair within a motion field (6.7).
- A rendering algorithm for density fields represented as blobs. This includes fast and consistent integration procedures (5.3) and a global illumination shooting algorithm (5.4).
- A physically motivated model for the spread of fire (6.4.2).
- Visual simulations for gases (6.7.1), fire (6.7.2) and hair (6.7.3) are achieved by combining the motion fields, blob simulation of the advection-diffusion equation and the rendering algorithms.

Following we briefly outline the notational conventions used throughout this thesis.

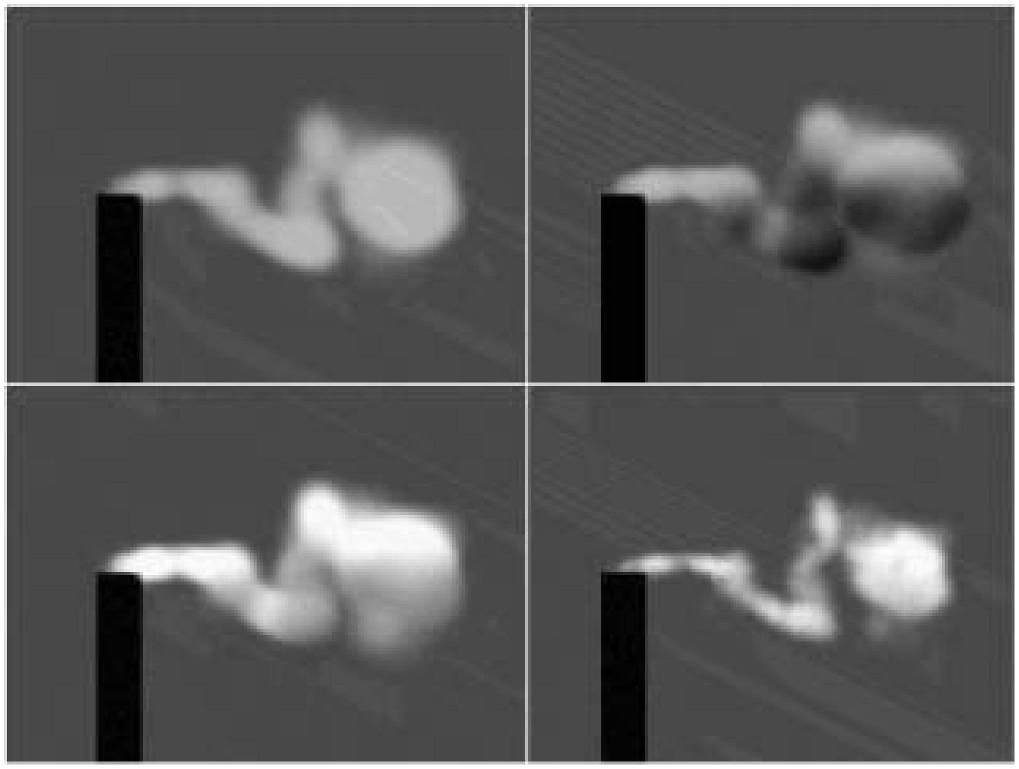

Figure 1.4: Four different types of rendering of the smoke. From top to bottom and left to right: (a) constant intensity, (b) self-shadowing, (c) shadowing and multiple scattering, (d) shadowing, multiple scattering and warping.

1.6 Notational conventions

We will use boldface to distinguish vectors and matrices from their components. Vectors are always denoted by lower case letters and are assumed to be in a column:

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \quad \text{and} \quad \mathbf{v}^T = (v_1, \cdots, v_n),$$

where "^{T}" denotes transposition. Upper case letters are reserved for matrices: $\mathbf{M} = (M_{ij})_{i=1,\cdots,n;j=1,\cdots,p}$. Using the column convention for vectors, matrix multiplication is done on the right side of the matrix, i.e.,

$$\mathbf{M}\mathbf{v} = (\mathbf{v}^T\mathbf{M}^T)^T.$$

The $n \times n$ identity matrix will be denoted by $\mathbf{I}_n$, and $\mathbf{v} = \mathbf{I}_n\mathbf{v}$. For complex vectors and matrices, the "*" symbol denotes transposition and complex conjugation of each element. The dot product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is written using the transpose operator:

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \sum_{i=1}^{n} u_i v_i.$$

In particular the norm of a vector can be denoted by

$$|\mathbf{v}|^2 = \mathbf{v}^T\mathbf{v}.$$

The tensor product between a vector $\mathbf{v}$ of dimension n and a vector $\mathbf{u}$ of dimension p is a $n \times p$ matrix denoted by

$$\mathbf{u}\,\mathbf{v}^T = (u_i v_j)_{i=1,\cdots,n; j=1,\cdots,p}.$$

The trace "tr" of a square $n \times n$ matrix $\mathbf{M}$ is defined as the sum of its diagonal elements:

$$\mathrm{tr}\mathbf{M} = \sum_{i=1}^{n} M_{ii}.$$

Very often a collection of vectors $\mathbf{v}_1, \cdots, \mathbf{v}_N$ will be considered and should not be confused with the components v_i of the single vector $\mathbf{v}$. A multi-dimensional function from $\mathbf{R}^d$ into $\mathbf{R}^m$ will be denoted by

$$\mathbf{f}(\mathbf{x}) = (f_1(x_1, \cdots, x_d), \cdots, f_m(x_1, \cdots, x_d))^T.$$

Uppercase letters will be reserved for functions that map $\mathbf{R}^d$ into the space of $m \times p$ matrices:

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} F_{11}(x_1, \cdots, x_d) & \cdots & F_{1m}(x_1, \cdots, x_d) \\ \vdots & & \vdots \\ F_{p1}(x_1, \cdots, x_d) & \cdots & F_{pm}(x_1, \cdots, x_d) \end{pmatrix}.$$

Similar notations are used when the set of real numbers $\mathbf{R}$ is replaced by the set of the complex numbers $\mathbf{C}$. The operator of differentiation ∇ is treated as a vector of (scalar) partial derivative operators

$$\nabla = \left(\frac{\partial}{\partial x_1}, \cdots, \frac{\partial}{\partial x_d}\right)^T.$$

Hence, the Laplacian, divergence and gradient of a function $\mathbf{f}$ are denoted respectively by:

$$\nabla^2 \mathbf{f} = \nabla^T \nabla \mathbf{f}, \qquad \sum_{i=1}^{d} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i} = \nabla^T \mathbf{f} \quad \text{and} \quad \left(\frac{\partial f_i(\mathbf{x})}{\partial x_j}\right)_{i=1,\cdots,m; j=1,\cdots,d} = \nabla \mathbf{f}^T.$$

The divergence of a function will also be denoted using the "dot" notation: $\nabla \cdot \mathbf{f}$. When the gradient is taken only with respect to a subset of the arguments it will be denoted by the partial notation. Let $\mathbf{z} = (x_1, \cdots, x_p)$ and $\mathbf{z}' = (x_p + 1, \cdots, x_d)$. The gradient with respect to the first p arguments will be denoted by:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{z}} = \left(\frac{\mathbf{f}(\mathbf{x})}{\partial x_1}, \cdots, \frac{\mathbf{f}(\mathbf{x})}{\partial x_p}\right).$$

1.7 Overview of the Thesis

The thesis is roughly divided into two parts. In Chapters 2 and 3 the technical machinery required for our models is presented. This exposition is interspersed with pointers to the actual simulations and with new material. These chapters are not essential in order to evaluate the contributions of this thesis. It was found that most physical models required for our simulations are instances of these general equations. For example, transport theory describes both the motion of liquids and the propagation of photons. In Chapter 2 we discuss various physical models of natural phenomena at different scales of detail. For each scale we give the general equations governing the physical system and review methods of solution. Specific examples of phenomena will be given. In Chapter 3 we review the theory of random functions on which stochastic modelling is based. The setting will be very general (random functions with multi-dimensional domain and values) but only relevant results will be mentioned. In Chapter 4 we present a general method of solving equations on an unordered set of points. The general methodology and existing techniques will be reviewed. In Chapter 5 a novel general method of solution for the intensity field in the presence of a participating medium is outlined. In Chapter 6 we describe various simulations of gaseous and related phenomena using the techniques explained in the previous sections. In Chapter 7 results and conclusions are stated.

Chapter 2

Particle Models For Natural Phenomena

We claim that our differential equations are 'true for every point' in the domain of a solution, and yet we introduce into these equations diffusion terms that treat of processes occuring at scales that are anything but infinitesimal-and this is especially so in the theory of turbulence. And yet it all succeeds, in that almost all scientists and engineers accept these concepts. Indeed, such acceptance nowadays constitute *an act of faith* in the truth of modern science, so that much matters come to appear as self-evident or self-revealed truths.
Michael Abbott

In this chapter we present physical models for both the propagation of light and for the dynamics of gases and fluids. Instead of presenting each model separately, we will derive them from a fundamental particulate description. Indeed, both light and gases are assumed to be composed of tiny fundamental particles. Under reasonable assumptions, the equations governing the motion of these particles are qualitatively the same. From the particulate description various physical models can be obtained at a succession of scales. At each approximation, some information about the full set of particles is discarded in exchange for a reduction in the size of the set of equations. We have chosen to present the models in this fashion to emphasize the existence of different physical models at different scales. Since the separation of scales is indeed one of the main themes of this thesis, it is therefore important to identify the physical model of a phenomenon relevant at a particular scale. Water, for example, can be modelled as a dynamical particle system of water molecules at the microscopic level, as a probability density of interacting elastic balls at the kinetic level or by a density field, a velocity field and a temperature field at the hydrodynamic level. The choice of a particular model depends on the scales which must be simulated. In computer graphics the relevant scales are determined by the realm of human perception. In this chapter we sketch how the various models are deduced from a particulate description using statistical mechanical arguments. Note that the emphasis will be on the phenomenology rather than on the technical derivation themselves. The latter can be found in the extensive literature on the subject matter, see e.g., [39].

2.1 Microscopic Level

2.1.1 Particle Systems

The most fundamental description of phenomena considered in this thesis is that of a *particle system*: a set of physical properties defined for N particles. For example, water can be modelled as a system of water molecules. The propagation of light can be approximated by the path of tiny elastic particles. This representation is different from the massless photon particle model used in quantum mechanics for example. The number N is very large in general, e.g., under normal pressure and temperature a liter of air contains on the order of $N = 10^{22}$ molecules. For our purposes the properties of each particle "i" are position $\mathbf{x}_i$, velocity $\mathbf{v}_i$ and mass m_i, with $i = 1, \cdots, N$. In order to simplify the derivations we assume that each particle has the same mass: $m_i = m$. The state of the particle at a certain time t is therefore defined by the following vector:

$$\mathbf{X}_N(t) = (\mathbf{x}_1(t), \mathbf{v}_1(t), \cdots, \mathbf{x}_N(t), \mathbf{v}_N(t)) \in \mathbf{R}^{6N}.$$

It is clear that this description excludes many physical phenomena. For example, the effects of polarization, diffraction and quantum effects are excluded from this description of the propagation of light. Indeed, the particulate formulation ignores the "wave-like" properties of light. The evolution over time of the particle system can then be given by the laws of Newtonian mechanics:

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i(t), \qquad \frac{d\mathbf{v}_i}{dt} = \frac{1}{m}\mathbf{F}_i \quad i = 1, \cdots, N. \tag{2.1}$$

supplemented with the initial conditions at $t = 0$:

$$\mathbf{X}_N(0) = (\mathbf{x}_1(0), \mathbf{v}_1(0), \cdots, \mathbf{x}_N(0), \mathbf{v}_N(0)).$$

The force exerted on particle i is the sum of an external force $\mathbf{F}_i^{ext}$, a force due to the boundary surfaces $\mathbf{F}_i^{bnd}$ and a term accounting for interparticle forces:

$$\mathbf{F}_i = \mathbf{F}_i^{ext} + \mathbf{F}_i^{bnd} + \sum_{j=1}^{N} \mathbf{F}_{ij}.$$

The interparticle force often is derived from a particle potential Φ:

$$\mathbf{F}_{ij} = -\nabla\Phi(\mathbf{x}_i - \mathbf{x}_j).$$

In general this potential is repulsive for short distances and attractive for longer distances. A simple potential which has these properties is the *Lennard-Jones* potential which solely depends on distance [28]:

$$\Phi_{LJ}(r) = \Phi_0 \left[\left(\frac{r_0}{r}\right)^{12} - 2\left(\frac{r_0}{r}\right)^{6} \right],$$

where r_0 is the radius beyond which the force becomes attractive and Φ_0 is the depth of the potential at r_0. The property of this potential is that the resulting force is symmetric around each particle. In general, however, the forces are asymmetric, such as in the case of water

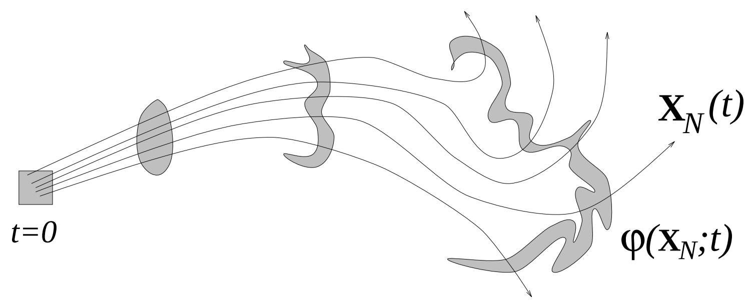

Figure 2.1: The Liouville Equation. The probability density of the particle system evolves in a manner similar to that of a fluid.

molecules. For some phenomena it is useful to decompose the velocity $\mathbf{v}_i$ of each particle into its direction $\mathbf{s}_i$ and its speed $v_i = |\mathbf{v}_i|$. This allows us to define the energy E_i of each particle in terms of its speed. For example, the kinetic energy of each particle i is

$$E_i = \frac{1}{2}mv_i^2.$$

Methods of Solution

The particle system can be simulated over time from a given initial state by integrating the mechanical equations stated in Equation 2.1. The most expensive component of each integration step is the calculation of the interparticle forces. By using a hierarchical data structure to store the particles, these forces can be computed in time $O(N \log N)$. More efficient algorithms running in time $O(N)$ have been developed for Coulombic potentials in two and three dimensions [24]. For potentials with a small support, the interparticle forces are calculated from nearby particles only. A linear time algorithm is obtained when the particles are stored in a grid and forces are calculated only between particles in neighbouring grid-cells only. This method is known as the *particle in cell* (PIC) method [28].

In computer graphics, particle systems have been used to model a wide range of phenomena. This approach is particularly appealing because of its simplicity. Systems of non-interacting particles have been used to model water, fire and grass among others uses [78, 93]. Interacting particle systems with a Lennard-Jones potential have been used to model simple viscous fluids [58]. Further, a potential which varies with temperature can be used to model phase transitions [95]. The inherent problem with these models is that the physical model is used at the wrong scale. A "particle" in these models is actually a blob of matter whose motion is not necessarily consistent with the equations of motion of microscopic particles. Furthermore, the Lennard-Jones potential provides a poor approximation of the asymmetric potential of water molecules. Hence, the resulting behaviour is itself an approximation to the real phenomenon.

2.1.2 Statistical Mechanics

Although the particle system formulation is very simple, it is impractical due to the large number N of particles needed for accurate simulations. Furthermore, the level of description is many orders of magnitude smaller than the scales of visual perception. When observing a gas, for example, we do not discern each individual molecule but rather variations in the density of gas molecules. Therefore we need to consider averages of the microscopic properties of the molecules. The technical apparatus needed to calculate averages of particle systems is given by *statistical mechanics*. Instead of deriving the equations for a single particle system, the goal of statistical mechanics is to derive equations for an *ensemble* of particle systems having the same probability density $\varphi(\mathbf{X}_N; t)$. In the language of probability theory, the statistical mechanical description of a particle system is a one-dimensional random function with values in a $6N$-dimensional space (see Section 3.1.2). The probabilistic description also arises quite naturally for practical reasons because the initial conditions of a system are never known exactly. Therefore an ensemble of particle systems starting in the same "volume of uncertainty" of initial conditions has to be considered. The state of the system after some time t is then only known approximately through its probability distribution. The evolution of a probability density function from an uncertain initial state is illustrated in Figure 2.1. Notice how the appearance of the density resembles that of a liquid immersed in a flow. In fact the *Liouville equation* governing the evolution of the probability density is a continuity equation similar to that found in fluid dynamics:[1]

$$\frac{\partial \varphi}{\partial t} = -\sum_{i=1}^{N} \left(\mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{x}_i} + \frac{\mathbf{F}_i}{m} \cdot \frac{\partial}{\partial \mathbf{v}_i} \right) \varphi.$$

This equation is in fact equivalent to Equation 2.1 which governs the motion of a single particle system. It is evident then that up to this point not much has been gained from the probabilistic framework. A reduced description of the system is obtained by defining marginal distributions of the probability density. The most elementary marginal distribution is the one-particle distribution given by:

$$\varphi_1(\mathbf{x}, \mathbf{v}; t) = N \underbrace{\int_{\mathbf{R}^6} \cdots \int_{\mathbf{R}^6}}_{N-1\ times} \varphi(\mathbf{X}_N; t)\, d\mathbf{X}_{N-1}.$$

This distribution gives the expected number of particles traveling with a velocity $\mathbf{v}$ at a point $\mathbf{x}$. The one-particle distribution can be used to define the following two physical quantities of the particles for example:

- The *density of mass* of the particles is defined as $\rho(\mathbf{x}, \mathbf{v}; t) = m\varphi_1(\mathbf{x}, \mathbf{v}; t)$.
- The *radiant intensity of light* is defined with respect to the one-photon distribution as $I_\nu(\mathbf{x}, \mathbf{v}; t) = (h\nu)c\varphi_1(\mathbf{x}, \mathbf{v}; t)$, where ν is the frequency, c is the speed of light and h is Planck's constant. The units of the intensity so defined are watts per unit area per unit solid angle in accordance with the radiometric definition of radiance [90].

An equation for the one-particle distribution is obtained by integrating the Liouville equation over the remaining $6(N-1)$ variables [28]:

$$\frac{\partial \varphi_1}{\partial t} + \mathbf{v}_1 \cdot \frac{\partial \varphi_1}{\partial \mathbf{x}_1} + \frac{1}{m}(\mathbf{F}_1^{ext} + \mathbf{F}_1^{bnd}) \cdot \frac{\partial \varphi_1}{\partial \mathbf{v}_1} = -\int_{\mathbf{R}^3} \int_{\mathbf{R}^3} \frac{\mathbf{F}_{12}^{col}}{m} \cdot \frac{\partial \varphi_2}{\partial \mathbf{v}_1}\, d\mathbf{x}_2\, d\mathbf{v}_2. \tag{2.2}$$

[1] $\partial/\partial \mathbf{x}_i = (\partial/\partial(x_1)_i, \partial/\partial(x_2)_i, \partial/\partial(x_3)_i)$, similarly for $\partial/\partial \mathbf{v}_i$.

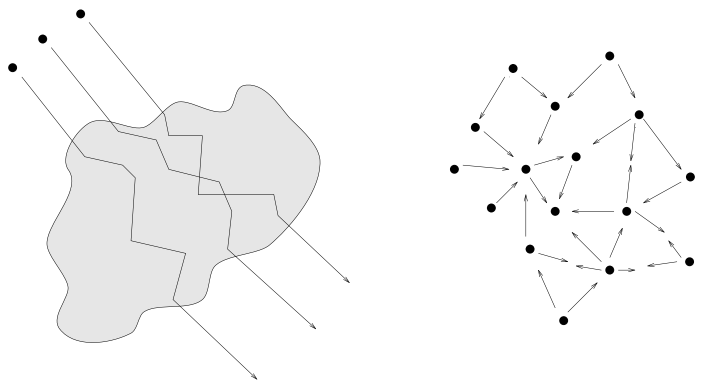

Figure 2.2: Two different types of collisions: particles streaming through a background medium versus particles interacting amongst themselves.

Although this equation looks complicated and cumbersome at first, it actually has a clear phenomenological interpretation. It states that the variation of the one-particle distribution over time along a given direction is due both to external forces and to interactions with all other particles. The *collision term* on the right hand side of the equation involves the distribution φ_2 which gives the expected product of the number of particles located at $\mathbf{x}$ and $\mathbf{x}'$ simultaneously and having velocities $\mathbf{v}$ and $\mathbf{v}'$ respectively:

$$\varphi_2(\mathbf{x},\mathbf{v},\mathbf{x}',\mathbf{v}';t) = N(N-1)\underbrace{\int_{\mathbf{R}^6}\cdots\int_{\mathbf{R}^6}}_{N-2\ times}\varphi_N(\mathbf{X}_N;t)\,d\mathbf{X}_{N-2}.$$

An equation similar to Equation 2.2 can be obtained for this distribution by integrating the Liouville equation over $6(N-2)$ variables. The resulting equation will involve the three-particle distribution φ_3. Likewise, an equation for φ_3 will involve a four-point particle distribution, and so on. The goal of *kinetic theory* is to introduce approximations into the theory in order to close this chain of equations.

2.2 Kinetic Theory

An understanding of the attitude of physicists towards thermodynamics and kinetic theory, is I think, to be sought only in the realm of psychology. *L. Bridgeman*

The equation for the one-particle distribution (Equation 2.2) cannot be solved directly since it involves the two-particle distribution which is *a priori* unknown. This term is therefore approximated in order to obtain a single equation for the one-particle distribution. As pointed out in the previous section, the term on the right hand side of Equation 2.2 characterizes the collisions between particles. It is therefore highly dependent on the type of phenomenon. In Figure 2.2 two extreme situations are illustrated. In the first case, the particles interact only with a participating medium but not amongst themselves. A good example of this is light particles streaming through a cloud. In the second case, the collision term is dominated by interparticle collisions, e.g., water molecules interacting with each other. Following, we consider the first case.

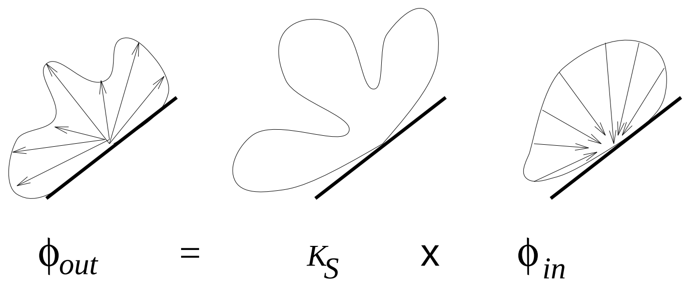

Figure 2.3: The boundary condition for the transport equation is given by the reflection probability distribution at the surfaces of the environment.

2.2.1 Transport Theory

In our derivation we make assumptions which are mostly relevant to the propagation of light. Firstly, we assume that the speed of each particle is constant and equal to v. The particle distribution is therefore a function of the position and direction of propagation only:

$$\phi(\mathbf{x}, \mathbf{s}, t) = \int_0^\infty \varphi_1(\mathbf{x}, r\mathbf{s}, t)\, dr.$$

Furthermore we assume that the particles do not modify the properties of the medium. Given these assumptions, the participating medium then changes the one-particle distribution by the following three "collision" events: a particle is absorbed, a particle is scattered into another direction or new particles are generated. Following the phenomenology of Equation 2.2, the variation of the one-particle distribution in a particular direction is equal then to a gain in particles due to in-scatter and emission minus a loss due to out-scatter and absorption. In particular, the collision term in this case is linear and equal to

$$-(\underbrace{vK_s(\mathbf{x})}_{\text{outscatter}} + \underbrace{vK_a(\mathbf{x})}_{\text{absorption}})\phi(\mathbf{x}, \mathbf{s}, t) + \underbrace{vK_s(\mathbf{x}) \int \Sigma(\mathbf{s}, \mathbf{s}')\phi(\mathbf{x}, \mathbf{s}', t)\, d\mathbf{s}'}_{\text{inscatter}} + \underbrace{vQ(\mathbf{x}, \mathbf{s}, t)}_{\text{emission}}.$$

The functions vK_a and vK_s describe the rate of absorption and the rate of scattering of the medium, respectively. Their sum $vK_t = vK_a + vK_s$ is the rate at which particles collide with the medium. The *scattering kernel* Σ gives the probability that the direction of a particle is changed from $\mathbf{s}'$ into $\mathbf{s}$ at a scatter event. The creation of new particles by the participating medium is entirely characterized by an emission distribution Q. Equation 2.2 with this collisional term becomes a linear equation:

$$\frac{1}{v}\frac{\partial \phi}{\partial t} + \mathbf{s} \cdot \frac{\partial \phi}{\partial \mathbf{x}} + \frac{1}{v}\frac{\mathbf{F}^{ext}}{m} \cdot \frac{\partial \phi}{\partial \mathbf{s}} + K_t \phi = K_s \int \Sigma(\mathbf{s}, \mathbf{s}')\phi(\mathbf{s}')\, d\mathbf{s}' + Q. \tag{2.3}$$

This equation is known as the *one-speed transport equation*, and is the fundamental equation for the propagation of light considered in this thesis [16]. In Chapter 5 we discuss the properties of the participating medium in more detail, presenting algorithms to solve this equation. To complete the mathematical description of the scattering equation, initial conditions and boundary conditions have to be specified. Initial conditions are specified by the distribution of particles at some initial time:

$$\phi_0(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}, \mathbf{s}, 0).$$

The boundary conditions are in general more complicated and depend on the problem at hand. Let S denote a boundary surface of the medium and $\mathbf{n}$ denote its normal function on this surface. For each point on the surface the boundary conditions can in general then be expressed as

$$\phi_{out}(\mathbf{x}, \mathbf{s}, t) = \int \Sigma_S(\mathbf{x}, \mathbf{s}, \mathbf{s}')\phi_{in}(\mathbf{x}, \mathbf{s}', t)(\mathbf{s}' \cdot \mathbf{n}(\mathbf{x}))\, d\mathbf{s}',$$

where the integral is over all directions $\mathbf{s}'$ such that $\mathbf{n} \cdot \mathbf{s}' < 0$ and Σ_S gives the reflection probability distribution of the surface. The distributions ϕ_{out} and ϕ_{in} denote the scattered and incident particle probability density, respectively. This situation is depicted in Figure 2.3.

Methods of Solution

Analytical solutions to the transport equation exist only for very restricted geometries. Therefore many numerical techniques have been developed to solve the transport equation. The most direct method is to discretize the domain of the particle probability density. A set of systems of equations is then obtained and can be solved using relaxation methods. Another method consists of expanding the particle probability density into some basis functions. Equations are then derived for the coefficients of the particle density. The P_N-equations are obtained if the basis functions dependence on direction are the spherical harmonics up to an order N. When local basis functions are used, a finite element approximation is obtained. These two methods are very memory intensive and can be used only for simple geometries or by making simplifying assumptions, such as, by assuming that the particle density is independent of the direction of the velocity. For arbitrary geometries, Monte-Carlo simulations are often used. In these simulations, the paths of sample particles are simulated through the environment according to the probability distribution given by the properties of the participating medium. From these samples an estimate of the particle density is obtained.

In computer graphics these algorithms have been applied to resolve the propagation of light in arbitrary environments. Discretizations have been used extensively to simulate diffuse environments with a constant participating medium [23]. An efficient relaxation scheme called "shooting" has been developed where the interaction from only one sample is considered at each iteration [14]. Angular dependence of the intensity field has been modelled by discretizing the direction field [29] or by adding a Monte-Carlo style pass to account for specular reflections [100, 91]. Discretizations in the presence of a participating medium have been considered for isotropic media [83] and arbitrary media, by discretizing the angles [43, 55]. Spherical harmonics expansions have been used separately for constant medium environments [92], and for gas densities with free boundary conditions [35]. The use of finite element solutions for environments with a constant media has been studied lately [107, 97]. Monte-Carlo simulations of the light transport have been explored, but these methods are of high computational cost [3, 69, 70].

2.2.2 Interacting Particle Systems

In interacting particle systems such as fluids, the collision term is usually non-linear and becomes more complicated. One of the most elementary formulations in this case is given by the *Boltzmann equation* which is obtained by considering only pair wise elastic collisions of

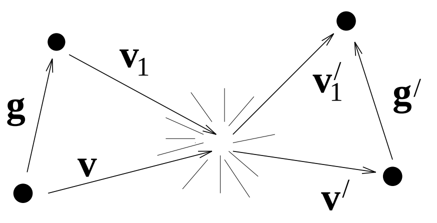

Figure 2.4: Collision between two particles. The collision depends only on the difference in velocities of the particles.

particles and by assuming that the probability of finding two particles at the same location $\mathbf{x}$ with different velocities $\mathbf{v}$ and $\mathbf{v}'$ is simply the product of the one-particle densities:

$$\varphi_2(\mathbf{x}, \mathbf{v}, \mathbf{x}, \mathbf{v}') = \varphi_1(\mathbf{x}, \mathbf{v}, t)\varphi_1(\mathbf{x}, \mathbf{v}', t).$$

The latter is known as the *molecular chaos* assumption. Consider the collision at a location $\mathbf{x}$ of two particles having velocities $\mathbf{v}$ and $\mathbf{v}_1$, respectively. After they collide the particles will have different velocities $\mathbf{v}'$ and $\mathbf{v}_1'$ as illustrated in Figure 2.4. Because the collisions are assumed to be elastic, they are entirely described by a collision kernel $\sigma(\mathbf{g}, \mathbf{g}')$ which depends only on the differences $\mathbf{g} = \mathbf{v} - \mathbf{v}_1$ and $\mathbf{g}' = \mathbf{v}' - \mathbf{v}_1'$. The rate of loss of particles along a direction $\mathbf{v}$ due to collisions is then given by:

$$|\mathbf{g}|\varphi_1(\mathbf{x}, \mathbf{v}, t)\varphi_1(\mathbf{x}, \mathbf{v}_1, t)\sigma(\mathbf{g}, \mathbf{g}')\ d\mathbf{v}_1\ d\mathbf{g}'.$$

A similar expression is obtained for the gain in particles along direction $\mathbf{v}$ due to collisions of the type $\mathbf{g}' \rightarrow \mathbf{g}$. By integrating the gain term minus the loss term over all incoming directions $\mathbf{v}_1$ and collision angles $\mathbf{g}'$ we get the total change due to collisions:

$$\int\int (\varphi_1(\mathbf{x}, \mathbf{v}_1', t)\varphi_1(\mathbf{x}, \mathbf{v}', t) - \varphi_1(\mathbf{x}, \mathbf{v}_1, t)\varphi_1(\mathbf{x}, \mathbf{v}, t))\, |\mathbf{g}|\sigma(\mathbf{g}, \mathbf{g}')\ d\mathbf{v}_1\ d\mathbf{g}'. \qquad (2.4)$$

Boundary conditions are similar to those given for the transport equation.

Methods of Solution

The Boltzmann equation is rarely used to simulate real flows. However, it is used to calculate parameters, such as the viscosity, that appear in hydrodynamic descriptions of a fluid (see next section). We mention that there exists a large body of literature which deals with solving the motion of a fluid by considering the interactions of particles on a discrete lattice. These techniques are broadly known as *Lattice Gas Automata* [7], and correspond to a discretization of the the Boltzmann equation. No applications of the Boltzmann equation in the field of computer graphics are known.

2.3 Macroscopic Level

Although kinetic equations can be used to model a wide range of phenomena, their solution becomes excessively complicated when collision events abound. In this case the velocity

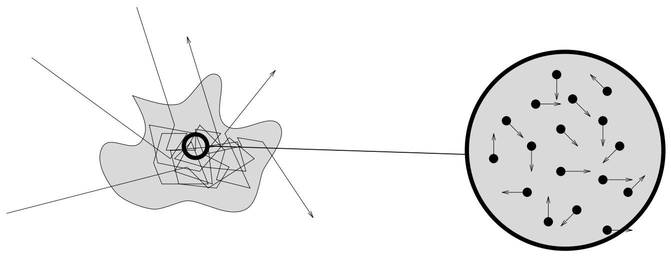

Figure 2.5: In the case of many collisions the distribution of the directions of the particles is nearly uniform.

of each particle changes very rapidly over time. When observing a small volume in space, we will find particles travelling in arbitrary directions (see Figure 2.5). The distribution of the direction of the velocities therefore becomes nearly uniform in systems dominated by collisions. Hence, the physical properties of the particle system become almost independent of velocity direction. The description of the particle system can be reduced further by removing the dependence on direction through integration. Indeed, let A be any property of the system depending on the velocity, then its *macroscopic average* over all directions is defined by:

$$\langle A\rangle_{dir}(\mathbf{x}, v, t) = \int A(\mathbf{x}, v\mathbf{s}, t)\, d\mathbf{s},$$

where v is the magnitude of the velocity. This averaging, for example, can be applied to both the linear transport equation and the non-linear Boltzmann equation. In the first case a diffusion equation is obtained and in the second case hydrodynamic equations are obtained. In both cases, the macroscopic averaging transforms the integral collision operator into a differential one. In other words when collision events dominate, the global description of the phenomenon collapses into a local one.

2.3.1 Diffusion Equations

From the one-particle distribution ϕ, the first two moments with respect to the velocity can be defined by the macroscopic averaging:

$$U(\mathbf{x}, t) = \langle\phi\rangle_{dir} \qquad \text{and} \qquad \mathbf{J}(\mathbf{x}, t) = \langle\phi\ \mathbf{s}\rangle_{dir}.$$

These two moments are known as the average density and the average flux of particles, respectively. Since the one-particle distribution in the case of many collision events has a smooth dependence on directions, we can assume that it is a function of these two moments only:

$$\phi(\mathbf{x}, \mathbf{s}, t) \approx \frac{1}{4\pi}U(\mathbf{x}, t) + \frac{3}{4\pi}\mathbf{J}(\mathbf{x}, t)\cdot\mathbf{s}.$$

This expansion is actually a truncated Taylor series in $\mathbf{s}$. An equation for the average density of particles can be obtained by first inserting this expansion into the transport equation (Equation 2.3). Taking the two first moments of this equation gives us two equations in U and $\mathbf{J}$ which can be combined to yield a single equation for the average particle density [16]:

$$\frac{1}{v}\frac{\partial U}{\partial t} - \nabla\cdot(\kappa\nabla U) + \frac{1}{v}\frac{\mathbf{F}^{ext}}{m}\cdot\nabla U + \alpha U = S. \qquad (2.5)$$

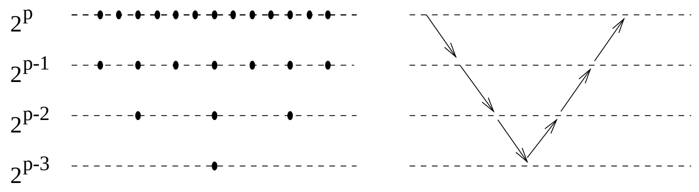

Figure 2.6: Different levels in the multi-grid method and a “v”-cycle.

This is the general form of an *advection-diffusion* equation. Such equations lie at the heart of most of the simulations considered in this thesis. The equation states that the evolution of U over time is characterized by diffusion at a rate κ, advection by an external force field $\mathbf{F}^{ext}$, absorption at a rate α and creation by sources S. This is a common transport mechanism which is often encountered in practice. Consider the evolution of a drop of milk in coffee. The milk is advected by the action of stirring, eventually diffusing with the coffee until a homogeneous mixture is obtained. We will use this equation to model the evolution of both the density and the temperature fields of a gas subjected to an external wind field. The equation will be used also to model the effects of multiple scattering in dense participating media. For this particular case we also give a more detailed derivation of the diffusion equation from the transport equation of light (see Section 5.5.2).

Method of Solutions

Advection-diffusion equations can be solved either by the finite difference or finite element method. The former is effective when the domain is two-dimensional or when the participating medium is nearly constant. For more general media the finite element method might provide tractable solutions.

Finite Differences

In finite differences, the average particle density is discretized into a set of discrete samples $\mathbf{U}_h(t) = (U_1(t), \cdots, U_N(t))$ defined on a regular grid of spacing $h \propto N^{-\frac{1}{d}}$, where $d =$ is the dimension of the space. The source term is discretized similarly into a set of samples $\mathbf{S}_h$. The diffusion, advection and absorption terms can be grouped into a single linear operator

$$\mathcal{L} = \nabla \cdot (\kappa \nabla) - \frac{1}{v} \frac{\mathbf{F}^{ext}}{m} \cdot \nabla - \alpha$$

and can be discretized into a matrix $\mathbf{M}_h$ using central differences [77]. One then obtains a discrete equation for the evolution of the sampled average diffuse intensity:

$$\frac{1}{v} \frac{\partial \mathbf{U}_h}{\partial t} = \mathbf{M}_h \mathbf{U}_h + \mathbf{S}_h.$$

The steady state solution of this equation is by iterating it from an initial value $\mathbf{U}_h(0)$ over discrete intervals of time Δt:

$$\mathbf{U}_h(k) = \mathbf{U}_h(k-1) + v\Delta t \left(\mathbf{M}_h \mathbf{U}_h(k-1) + \mathbf{S}_h \right), \qquad k = 1, 2, 3, \cdots.$$

The above iteration scheme is known as the *Jacobi method* and converges when the matrix $\mathbf{M}_h$ is diagonally dominant. Unfortunately, it converges slowly. A powerful technique to speed up the convergence rate is to relax the system on grids with different spacings h. This method is known as the *multi-grid method* and is briefly reviewed next (for more details see for example [25]). The efficiency of the multi-grid method is due to both the fact that it can produce a good initial estimate of the solution and the fact that it removes the high frequencies from the error by relaxing on coarser grids. These are achieved by considering a hierarchy of grids of spacings $h = 2^p$, $p = p_{coarse}, \cdots, p_{fine}$. The equation first is relaxed on the finest grid for a fixed number of iterations and then projected onto the next coarsest grid. This projection is likewise relaxed for a fixed number of iterations. These two steps are repeated until the coarsest level has been reached. The whole process is then reversed: each approximation is interpolated and relaxed on to the next finer grid. This process is repeated until the finest grid is reached. The whole procedure just described corresponds to a complete "v-cycle" as illustrated in Figure 2.6. An approximation of the solution is obtained by going through a fixed number of such v-cycles until convergence. This results in speed-ups of an order of magnitude faster than simple relaxation. However, this method is very memory intensive for three-dimensional domains. The method of finite elements described next attempts to resolve this problem.

Finite Elements

In finite element methods the average particle density is discretized by an expansion into a set of basis functions (elements) $\{\psi_i\}_{i=1}^N$:

$$U(\mathbf{x}, t) = \sum_{i=1}^{N} U_i(t)\psi_i(\mathbf{x}).$$

The source intensity is expanded likewise. The functions are usually chosen such that they are close to an orthogonal basis, in the sense that for each function ψ_i the product

$$((\psi_i, \psi_j)) = \int \psi_i(\mathbf{x})\psi_j(\mathbf{x})\, d\mathbf{x} \tag{2.6}$$

is non-zero for only a few functions ψ_j. A linear equation for these coefficients U_i can be obtained for the steady state solution ($\frac{\partial U}{\partial t} = 0$ in Eq. 2.5) by many means. First the expansions of the average diffuse intensity and the source term are substituted into the steady state diffusion equation:

$$\sum_{i=1}^{N} U_i \mathcal{L}\psi_i(\mathbf{x}) - S_i\psi_i(\mathbf{x}) = 0. \tag{2.7}$$

In a *collocation method*, N linear equations are obtained by requiring that this equation be satisfied at N distinct points $\mathbf{x}_1, \cdots, \mathbf{x}_N$ [64]. The linear system can then by solved directly by a matrix inversion which is stable when N is not too large and the points are equi-distributed over the domain of interest. In the *Galerkin method* a set of linear equations is obtained by taking the product $((\cdot, \psi_j))$ (see Equation 2.6) on both sides of Equation 2.7:

$$\sum_{i=1}^{N} U_i((\mathcal{L}\psi_i, \psi_j)) - S_i((\psi_i, \psi_j)) = 0, \qquad j = 1, \cdots, N.$$

In the event that the above products can be calculated, an approximation of the solution can be calculated by direct matrix inversion. In both methods, the boundary conditions can be satisfied naturally if each basis function ψ_i fulfills them. For self-adjoint and positive definite operators $\mathcal{L}$, the two methods outlined above converge when the basis functions are *complete*: any smooth function can be approximated arbitrarily closely by a linear combination of the basis functions [64]. For example, if the advecting force is null, then the linear operator $\mathcal{L} = \nabla\kappa\nabla - \alpha$ and is self-adjoint and positive definite.

2.3.2 Hydrodynamic Equations

The Navier-Stokes equations are the most idealized statements possible about a physically non-existent situation. *Michael Abbott*

Deriving equations for interacting particle systems using the macroscopic averaging scheme $\langle\rangle_{vel}$ is considerably harder due to the non-linearity of the Boltzmann collision term (recall Equation 2.4). There is however a general procedure to obtain macroscopic equations in terms of physical quantities that are conserved during a collision. For the case of interacting point particles these quantities are mass m, momentum $m\mathbf{v}$ and local kinetic energy $\frac{1}{2}m|\mathbf{v}-\mathbf{u}|^2$. The resulting fields corresponding to these conserved quantities are the density ρ, velocity $\mathbf{u}$ and local temperature T of the particle system, respectively. Equations for these fields can be derived directly from the Boltzmann equation using the so called *Chapman-Enskog expansion* [39]. If only first-order terms are considered, the usual *Navier-Stokes Equations* are obtained [16]:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{2.8}$$

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)\mathbf{u} - \frac{\rho\mathbf{F}^{ext}}{m} = -\nabla p + \nu\nabla^2\mathbf{u} \tag{2.9}$$

$$\rho C_p\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)T = -\rho(\nabla\cdot\mathbf{u})T + \nabla\cdot k\nabla T. \tag{2.10}$$

where ν, C_p and k are averages of functions which depend on the three collisional invariants. They have the following physical meaning: ν models the viscosity of the fluid, C_p is the specific heat of the fluid and k is the temperature conductance. The pressure p is an independent variable which can be removed from the set of equations in case the fluid is *incompressible*, i.e., when the density of the fluid is constant, Equation 2.8 reduces to:

$$\nabla\cdot\mathbf{u} = 0.$$

This condition is satisfied approximately even for fluids having a non-constant density when the velocity of the fluid is considerably smaller than the speed of sound [47]. In case the velocity field and the density is known, Equation 2.10 becomes a linear advection-diffusion type equation. We have used this approach to model the evolution of gaseous phenomena such as fire and steam (see Chapter 6).

Methods of Solution

As for the diffusion equation, approximate solutions for the hydrodynamic equations can be obtained by discretizing the spatial and temporal domains. However, the application of these methods to the hydrodynamic equations is more difficult because of the non-linearities and the large range of scales which have to be modelled. Simulations can be modelled for large values of the viscosity which diminish the number of scales and therefore the resolution of the computational grid. Smaller scales are modelled by a statistical model and are incorporated into a high viscosity simulation. The latter simulations are known as *Large Eddie Simulations* [81].

In computer graphics, many researchers have used the hydrodynamic equations to model simple fluids. The evolution of ocean waves has been modelled using a linearized version of these equations [52]. This model was supplemented by an ad hoc model for the simulation of wave-breaking [19]. A similar linearization of the Navier-Stokes equation was used to model viscous fluids responding to external forces [37].

Chapter 3

Stochastic Modelling

Everything is approximate, less than approximate, for when more closely and sharply examined, the most perfect picture is a warty, threadbare approximation, a dry porridge, a dismal mooncrater landscape. What arrogance is concealed in perfection. Why struggle for precision, purity, when they can never be attained? The decay that begins immediately on completion of the work was now welcome to me. *Jean Arp*

Most natural phenomena exhibit a high degree of complexity. In practice this irregularity is difficult to simulate using only physical models. For example, a hydrodynamic description of a fluid gives rise to extremely complicated simulations of flows when the viscosity is sufficiently low. However, humans can effectively classify and describe highly irregular phenomena. We can recognize a pine tree from a maple tree, for example, even though both have complicated shapes, because both have very different global appearances. Within a particular class of tree such as pine trees, however, differentiation becomes more difficult and is achieved only by pointing to local detail dissimilarities. In short, although all trees have unique complicated shapes, we are able to describe and differentiate them by their global shape and texture. Stochastic modelling is a formalization of descriptions at this level of detail. A stochastic model does not define a particular shape but rather defines a class of shapes which have the same global structure and texture. The precise shape will remain unknown until a particular realization of the stochastic model is observed or computed. Stochastic modelling therefore is a natural framework for the design of complicated shapes and the animation of irregular phenomena because the complexity of a specific realization can be generated by an automatic process.

The justification for the use of stochastic models over deterministic models is entirely practical. Even though the underlying physical principles are deterministic, we use stochastic models in order to get tractable solutions.

This chapter describes both background material and new algorithms. The first section is devoted to discussing the main concepts and results of the probability theory of random functions. The second section presents a general methodology to obtain new random functions from existing ones using general transformations. In the third section we present various new and existing algorithms to synthesize realizations of random functions. The last section applies these results to the synthesis of turbulent wind fields.

3.1 Probability Theory of Random Functions

This section will present results and concepts from probability theory needed in order to understand the new algorithms developed in this thesis. The presentation is not intended to be a tutorial on the probability theory of random functions, since a comprehensive survey of this vast subject would not fit in this thesis. Moreover, excellent text books exist on the matter, ranging from purely mathematical to totally application oriented. The former may appeal to the mathematician since each concept and result is translated into a rigorous definition and theorem. The latter text books are intended for the engineer with a specific application in mind. In particular the application oriented books spend alot of time giving the reader an intuitive feel for the concepts and results. The presentation given in this thesis is written with computer graphics applications in mind. Of course, attention has been given to ensure equations and definitions are correct and sound. The first section of this chapter is a condensed version from material gleaned from various sources. Some of these sources are briefly reviewed. The mathematical reference to the theory of random functions are the three volumes by the Russian mathematicians Gikhman and Skohorod [21]. All the theorems cited in this thesis are proven in their work. An excellent text which interweaves mathematical proofs with practical applications is the French book written by Krée and Soize [41]. A very concise and pedagogical presentation of the fundamentals of the theory is given in the first part of Panchev's book on turbulence [67]. Yaglom's text book on the theory of homogeneous random functions is probably the best text book for someone interested in both the applications and the underlying mathematical proofs [106]. The latter are given in the second volume, in note form. These texts generally begin by presenting the theory for scalar valued random functions depending on time only and then move on to generalize all the concepts to higher dimensions. In this thesis the presentation will be set in arbitrary dimensions from the beginning. We will use a compact vector notation to keep the equations more accessible.

3.1.1 Random Variables

No two days are alike, nor even two hours, and ever since the creation of the world no two leaves on a tree are the same. *John Constable*

Many events that occur around us have a behaviour which we cannot predict with absolute certainty. Two such events are winning the lottery, and dying within a hundred years. Although both events are uncertain, we can be fairly confident that the former is almost impossible, with the latter being almost certain. In fact, the chances of winning the lottery can be explained more precisely using probability theory. Hence, we would like to assign a number between 0 (impossible) and 1 (certain) to events that surround us. Of course not everyone will agree on these numbers. It would be extremely difficult for a mathematician and an avid lottery player to agree on the chances of winning, for example. Probability theory begins with an event space $\mathcal{E}$ and a probability measure P. The probability measure is not defined over the event set but rather over a set $\mathcal{B}$ of subsets of events. This is because we want to assign probabilities not only to single events but also to combinations of events, such as "it rains *and* I forgot my umbrella". The set of subsets of events cannot be arbitrary and has to be "big enough" to be useful in probability theory. More precisely, it has to satisfy the following conditions:

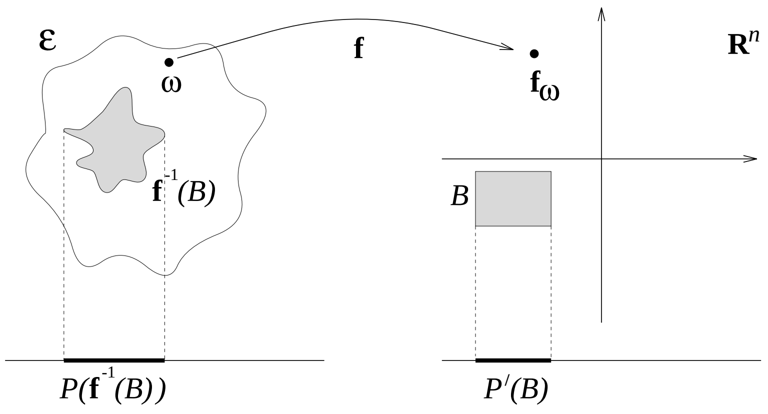

Figure 3.1: The probability measure on $\mathbf{R}^n$ is obtained via the random variable $\mathbf{f}$.

(1) $\mathcal{B}$ contains the empty set and the whole event space.
(2) $\mathcal{B}$ is invariant under complements, e.g., if we know the probability of the event "it will rain tomorrow" then automatically we know the probability that it won't rain tomorrow.
(3) The union of a countable number of elements in $\mathcal{B}$ has to remain in $\mathcal{B}$.

A set having these properties is called a $\sigma - algebra$ by mathematicians. Given these properties we can define a probability measure P from the space $\mathcal{B}$ into the unit interval $[0, 1]$. The only constraint on the probability measure is that the probability of a countable union of disjoint events is equal to the sum of the probabilities of each single event. In short, if we want to use the results of probability theory, we have to make sure that our space of subspaces of events is a σ-algebra and that our probability measure has the property just given. To package the whole description, the triplet $(\mathcal{E}, \mathcal{B}, P)$ is called a *probability space.*

Often when dealing with random events we would like to answer such quantitative questions as "what is the most likely event?" or "how are the events spread around the most likely event?" In order to answer such questions one usually has to perform operations and computations on events. In order for this to be possible, we have to assign numerical values to each event. A *random variable* $\mathbf{f}$ which maps the event space into the space $\mathbf{R}^n$ is introduced into the theory for exactly these reasons:

$$\mathbf{f} : \begin{cases} \mathcal{E} \longrightarrow \mathbf{R}^n \\ \omega \longmapsto \mathbf{f}_\omega \end{cases}$$

Each value $\mathbf{f}_\omega$ of the random variable $\mathbf{f}_\omega$ is called a *realization.* The probability measure P induces both a probability measure $P_{\mathbf{f}}$ and a σ-algebra $\mathcal{B}_{\mathbf{f}}$ on $\mathbf{R}^n$ via the random variable (see Figure 3.1):

$$\begin{aligned} &B \in \mathcal{B}_{\mathbf{f}}, \quad \text{if } \mathbf{f}^{-1}(B) \in \mathcal{B} \quad \text{and} \\ &P_{\mathbf{f}}(B) = P\left(\mathbf{f}^{-1}(B)\right). \end{aligned}$$

With this measure thus defined we can associate a *distribution function* $F_{\mathbf{f}}$ defined over the space $\mathbf{R}^n$. Indeed for each element $\mathbf{z}$ one can define the set of all elements whose coordinate values are smaller than $\mathbf{z}$ componentwise (see Figure 3.2). The value of the probability distribution at a point $\mathbf{z}$ is given by the probability of this set:

$$F_{\mathbf{f}}(\mathbf{z}) = P_{\mathbf{f}}(\{\mathbf{z}' \in \mathbf{R}^n \mid z'_1 < z_1 \text{ and } z'_2 < z_2 \text{ and } \cdots \text{ and } z'_n < z_n\}).$$

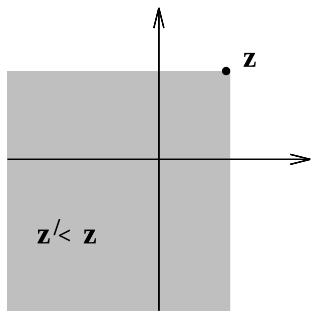

Figure 3.2: Definition of the probability distribution.

In the case when this distribution can be factored into

$$F_{\mathbf{f}}(\mathbf{z}) = F_{f_1}(z_1) \cdots F_{f_n}(z_n),$$

the components are said to be *independent*. The probability of any subset B of events in $\mathcal{B}_{\mathbf{f}}$ in terms of the distribution function F is then given by a *Stieltjes* integral. For a d dimensional hyper-box R, the definition is:

$$P_{\mathbf{f}}(R) = \int_R dF_{\mathbf{f}}(\mathbf{z}) = \lim_{h \to 0} \sum_{\mathbf{l} \in R_h} \sum_{i=1}^{d} F_{\mathbf{f}}(\mathbf{z}_{\mathbf{l}}) - F_{\mathbf{f}}(\mathbf{z}_{\mathbf{l}-\mathbf{e}_i}), \tag{3.1}$$

where the samples $\mathbf{z}_{\mathbf{l}}$ are defined on a regular discretization R_h of the hyper-box R with spacing proportional to h and the $\mathbf{e}_i$ are the unit vectors of $\mathbf{R}^d$. The definition for an arbitrary domain B follows from the fact that any domain can be approximated by a union of hyper-boxes. A Stieltjes integral is a generalization of the standard Riemann integral. To see why this is so, suppose that the probability distribution is differentiable. This allows us to define a *probability density function* $\varphi_{\mathbf{f}}$ by:

$$dF_{\mathbf{f}}(\mathbf{z}) = \varphi_{\mathbf{f}}(\mathbf{z}) d\mathbf{z}.$$

Intuitively, the probability density gives the probability of a small set of events centred at $\mathbf{z}$. In this case the Stieltjes integral of Equation 3.1 is actually a Riemann integral. However, for general probability distributions it is necessary to use a Stieltjes integral. Via the random variable $\mathbf{f}$ we can thus forget about the original probability space and work solely with the new probability space $(\mathbf{R}^n, \mathcal{B}_{\mathbf{f}}, P_{\mathbf{f}})$. A *stochastic model* for a phenomenon is the specification of such a probability space. The determination of a stochastic model for a particular phenomenon is called *stochastic modelling* [41]. To relate the stochastic model of a phenomenon to observations on it, the concept of averages is introduced. This is because averages are usually more intuitive and easier to estimate from observed realizations of a real phenomenon. For example, the average temperature at noon in downtown Toronto can be estimated by measuring the noonday temperature for many consecutive days and taking the arithmetic mean. In addition the average fluctuation around this mean can be calculated by taking the arithmetic mean of squared differences between the measurements and the estimated mean. More generally, it is useful to define the average of an arbitrary function of the random variable. Let $\mathbf{h}$ be a function from $\mathbf{R}^n$ into some other space $\mathbf{R}^k$, the *average* of this function applied to the random variable $\mathbf{f}$ is defined by

$$\langle \mathbf{h}(\mathbf{f}) \rangle = \int_{\mathbf{R}^n} \mathbf{h}(\mathbf{z}) \; dF_{\mathbf{f}}(\mathbf{z}),$$

and is an element of the vector space $\mathbf{R}^k$. Intuitively, it is the sum of all possible realizations of the random variable weighted by their respective probabilities (the intuition is exact in the case where the probability space is discrete). Because continuous functions can be approximated arbitrarily closely by polynomials, an important class of averages of a random variable are its *moments*. The moments of order p are defined as:[1]

$$\langle \mathbf{f}^{\mathbf{a}} \rangle = \int_{\mathbf{R}^n} \mathbf{z}^{\mathbf{a}} \; dF_{\mathbf{f}}(\mathbf{z}), \qquad \mathbf{a} \in \mathbf{N}^n \text{ and } |\mathbf{a}| = p.$$

Indeed the class of random variables with finite moments of order two have *finite energy* and form a Hilbert space with a scalar product given by the probability distribution.

$$\mathbf{f} \cdot \mathbf{g} = \langle \mathbf{f}^T \mathbf{g} \rangle = \int_{\mathbf{R}^n} \int_{\mathbf{R}^n} \mathbf{z}^T \mathbf{z}' \; dF_{\mathbf{g}}(\mathbf{z}') \; dF_{\mathbf{f}}(\mathbf{z}).$$

In particular this scalar product defines a norm. Hence one can give a meaning to the limit $\mathbf{f}$ of a sequence of random variables $\mathbf{f}_1, \mathbf{f}_2, \cdots$ when

$$\lim_{k \to \infty} \langle |\mathbf{f}_k - \mathbf{f}| \rangle$$

exists. In other words the sequence converges to the limit in the mean or "most of the time". The limit is also often denoted by $\text{l.i.m}_{k\to\infty}\mathbf{f}_k$, where l.i.m stands for *limit in the mean*. In addition, random variables with finite energy have the property of having a finite mean with their *correlation* $\mathbf{C}_{\mathbf{f}}$ being well defined:

$$\mathbf{C}_{\mathbf{f}} = \langle \mathbf{f} \; \mathbf{f}^T \rangle = \int_{\mathbf{R}^n} \mathbf{z}\mathbf{z}^T \; dF_{\mathbf{f}}(\mathbf{z}).$$

This matrix in effect contains all moments of order 2 of the random variable. It quantifies the relatedness between each pair of components of the random variable $\mathbf{f}$. In particular if the correlation is zero the pair of components are *uncorrelated*. By definition the correlation is symmetric and positive definite. Two random variables $\mathbf{f}$ and $\mathbf{g}$ are *uncorrelated* when their respective components are uncorrelated:

$$\langle \mathbf{f} \; \mathbf{g}^T \rangle = \mathbf{0}.$$

We now give two concrete examples of a probability distribution. The *uniform distribution U* defines one of the simplest probability spaces. This distribution is constant on a subdomain B of $\mathbf{R}^n$ and zero elsewhere:

$$dU(\mathbf{z}) = \begin{cases} \frac{1}{vol(B)} d\mathbf{z}, & \text{if } \mathbf{z} \in B \\ 0, & \text{elsewhere.} \end{cases}$$

Both its mean and its correlation function depend on the domain B. Another important distribution with which most people are familiar is the bell shaped Gaussian distribution.

[1]If $\mathbf{a} = (a_1, \cdots, a_n) \in \mathbf{N}^n$, then a vector $\mathbf{z}^{\mathbf{a}}$ is defined as $z_1^{a_1} z_2^{a_2} \cdots z_n^{a_n}$ and $|\mathbf{a}| = a_1 + a_2 + \cdots + a_n$.

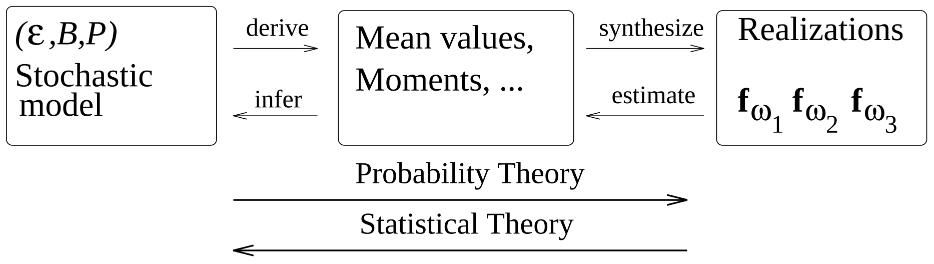

Figure 3.3: Probability theory proceeds conversely from the statistical theory.

Many phenomena (typically deviations) follow this distribution very closely. A generalization of the one-dimensional Gaussian bell distribution is given by

$$dG(\mathbf{z}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det \mathbf{C}}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m})^T \mathbf{C}^{-1}(\mathbf{z}-\mathbf{m})\right) d\mathbf{z}, \tag{3.2}$$

where $\mathbf{m}$ is an arbitrary vector and $\mathbf{C}$ is a positive definite symmetric matrix with a non-zero determinant $\det\mathbf{C}$. It can be proven that a random variable having this Gaussian probability distribution has a mean of $\mathbf{m}$ and correlation of $\mathbf{C}$. This implies that the mean and the correlation are sufficient to determine a Gaussian distribution entirely. Generally, this is not true for most distributions which are described by a larger (possibly infinite) set of moments. In fact a sum of uncorrelated random variables having the same distribution tends towards a Gaussian distribution as the number of random variables increases. This result is known as the *central limit theorem*.

We have, up to this point presented only the probability theory of random variables: that from a definition of a probability space various concepts and constructs are derived, such as the mean and the correlation. The *statistical theory* of random variables proceeds conversely with respect to probability theory. From a given set of observables, estimates of the mean values and moments are computed, from which a probability space is inferred. Usually a given family of probability distributions is postulated whose parameters are then estimated from the observables. Additional statistical tests can be applied to test the "goodness of fit". In this thesis we follow the path of probability theory, since our goal is to derive stochastic models and synthesize data. The difference between the two theories is illustrated in Figure 3.3.

3.1.2 Random Functions

If you turn on the radio you will immediately know if you are hearing classical music or the Beatles. And if you have the slightest interest in classical music you can distinguish Bach from sixteenth-century music, Beethoven from Bach, and Bartok from Beethoven. You may not have heard the pieces before, but there is something unique about the arrangement of sounds that allows almost instant recognition. One can try to capture this "something unique" by statistical studies. *David Ruelle*

Most natural phenomena are more conveniently modelled by a function rather than by a single variable. Clouds, for example, can be modelled by a density function which varies in space and in time. The concept of a random variable therefore has to be extended to that of a *random function* which maps an event space into a space of functions:

$$\mathbf{f} : \begin{cases} \mathcal{E} \longrightarrow \mathcal{C}(d,m) \\ \omega \longmapsto \mathbf{f}_\omega \end{cases}$$

In this case each realization is an element of the space of continuous functions $\mathcal{C}(d,m)$ mapping elements of $\mathbf{R}^d$ into $\mathbf{R}^m$. The space $\mathbf{R}^d$ will be called the *spatial domain* of the random function. To make the dependence on the dimension explicit, the random function is referred to as a *d-dimensional random function with values in an m-dimensional space.* Next, we give some examples of phenomena which can be modelled by a random function. Many different values for the dimensions d and m arise demonstrating the need for a presentation of the theory of random functions set in arbitrary dimensions.

- **Terrain.** Natural terrain with no foldover structures can be modelled as a random height function $h(x,y)$ assigning an elevation to each point (x,y) of the plane. In this case $d=2$ and $m=1$.

- **Ocean Waves.** The temporal evolution of the surface of the ocean can be modelled by adding a temporal dependence to the surface terrain model: $s(x,y,t)$. In this case $d=3$ and $m=1$.

- **Clouds.** The evolution of a density field of cloud water droplets in space can be modelled by a four-dimensional random function $\rho(x,y,z,t)$. In this case $d=4$ and $m=1$.

- **Wind Fields.** An evolving wind field can be modelled by a random function which assigns a three-dimensional vector to each point in space-time: $\mathbf{u}(\mathbf{x},t)$. In this case $d=4$ and $m=3$.

- **Mechanical Structures.** The kinetics of a mechanical structure defined by the positions of N point masses can be modelled by a random function $\mathbf{f}(t) = (\mathbf{x}_1(t), \cdots, \mathbf{x}_N(t))$. In this case $d=1$ and $m=3N$.

- **Particle Systems.** The dynamics of a system of N particles having identical mass can be modelled by a random function $\mathbf{f}(t) = (\mathbf{x}_1(t), \mathbf{v}_1(t); \cdots; \mathbf{x}_N(t), \mathbf{v}_N(t))$. In fact this is the fundamental description used in Statistical Mechanics (see previous chapter). In this case $d=1$ and $m=6N$.

Although the definition of a random function is analogous to that of a random variable, the results outlined in the previous section cannot be applied directly to random functions. The reason for this is that the space of all continuous functions is infinite dimensional, therefore such spaces require more sophisticated mathematical tools. In practice, however, a random function can be reduced to a random variable by sampling it at N points $\mathbf{x}_1, \cdots, \mathbf{x}_N$. The sampled random function is then equivalent to a single random variable with values in $(\mathbf{R}^m)^N$:

$$\begin{cases} \mathcal{E} \longrightarrow \mathbf{R}^m \times \cdots \times \mathbf{R}^m \\ \omega \longmapsto (\mathbf{f}_\omega(\mathbf{x}_1), \cdots, \mathbf{f}_\omega(\mathbf{x}_N)) \end{cases}$$

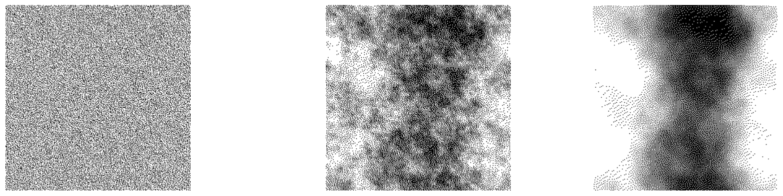

Figure 3.4: Random functions with different spatial structure: white noise (left), slightly correlated (middle) and highly correlated (right).

For each number N of samples we obtain a different probability distribution F_N called the *N-point probability distribution* of the random function. These probability distributions depend in general on the location of the samples and are therefore written as

$$F_N : \left\{ \begin{array}{l} (\mathbf{R}^d)^N \times (\mathbf{R}^m)^N \longrightarrow [0,1] \\ (\mathbf{x}_1, \cdots, \mathbf{x}_N; \mathbf{z}_1, \cdots, \mathbf{z}_N) \longmapsto F_N(\mathbf{x}_1, \cdots, \mathbf{x}_N; \mathbf{z}_1, \cdots, \mathbf{z}_N) \end{array} \right.$$

An approximate description of a random function is therefore given by a finite number of probability distributions F_N. These distributions permit us to calculate average values of the random function. The one-point probability distribution is used to define the mean of the random function:

$$\bar{\mathbf{f}}(\mathbf{x}) = \langle \mathbf{f}(\mathbf{x}) \rangle = \int_{\mathbf{R}^m} \mathbf{z}\, dF_1(\mathbf{x}; \mathbf{z}).$$

The mean is a deterministic function in the functional space $\mathcal{C}(d, m)$ and is obtained by averaging over all possible realizations of the random function. This method of defining averages is known as *statistical averaging* and should not be mistaken with *spatial averaging.* In the latter the mean of the random function is calculated by averaging the values of a particular realization $\mathbf{f}_\omega$ of the random function over its domain:

$$\langle \mathbf{f} \rangle_\omega = \lim_{k \to \infty} \frac{1}{vol(V_k)} \int_{V_k} \mathbf{f}_\omega(\mathbf{x})\, d\mathbf{x}, \quad V_k \to \mathbf{R}^d \text{ as } k \to \infty.$$

Spatial averaging is peculiar to random functions and is often used to compute average values from observed data. For example, the mean wind velocity at a point in space is usually calculated by taking a time average of the measurements. In particular, spatial averaging depends on the choice of the realization and is always equal to a constant. In general different constants are obtained for different choices of the realization. The random function is called *ergodic* when both averaging methods coincide. In particular, the mean $\bar{\mathbf{f}}$ obtained by statistical averaging has to be constant and the spatial average $\langle\rangle_\omega$ has to be the same for each realization. Ergodicity is important in the statistical theory of random functions as it relates spatial averages obtained from measurements to statistical averages. In practice it is hard to determine whether a certain phenomenon can be modelled by an ergodic random function. Very often the ergodic hypothesis is introduced as a matter of convenience without further justification. In this thesis we consider only statistical averages and we consequently do not invoke the ergodic hypothesis.

The mean only characterizes the average value of the random function at different points. It does not tell us anything about the spatial structure of the random function. In Figure

3.4 we show three random functions having the same mean equal to 128. The two random functions on the right of Figure 3.4 clearly have more structure than the random function on the left. Spatial structure is characterized by probability distributions involving more than one sample. The simplest of these is the two point probability distribution F_2 which permits us to define the amount of correlation between the values of the random function evaluated at two different points:

$$\mathbf{C_f}(\mathbf{x},\mathbf{x}') = \langle \mathbf{f}(\mathbf{x})\mathbf{f}^T(\mathbf{x}')\rangle = \int_{\mathbf{R}^m}\int_{\mathbf{R}^m} \mathbf{z}_1\mathbf{z}_2^T \; dF_2(\mathbf{x},\mathbf{x}';\mathbf{z}_1,\mathbf{z}_2).$$

The correlation function is in fact a tensor product of the random function evaluated at two different points. Similarly, it is possible to define higher order tensors using probability distributions involving more than two samples. This is not necessary when both the one point and two point probability distributions are Gaussian (see Equation 3.2). This implies that the remaining N-point probability distributions are entirely determined by the one and two point probability distributions. In the remainder of this thesis we will be concerned only with random functions having Gaussian N-point probability distributions. This condition does not always hold for natural phenomena. Indeed, probability distributions estimated from experimental data were found to be non-Gaussian [47]. However, the theory for non-Gaussian random functions is considerably more complicated and does not lend itself easily to analytical investigations. This assumption is usually invoked for purely practical reasons. For our purposes, a stochastic model for a phenomenon is thus entirely specified by the mean and the correlation function only. The structure of a random function can be characterized by functions other than the correlation function. In fact a more intuitively appealing characterization is given by the *structure function*:

$$\mathbf{D_f}(\mathbf{x},\mathbf{x}') = \langle (\mathbf{f}(\mathbf{x}')-\mathbf{f}(\mathbf{x}))(\mathbf{f}(\mathbf{x}')-\mathbf{f}(\mathbf{x}))^T\rangle$$

This function essentially measures the absolute difference between the values of the random function at two different points. In particular, for structured random functions, this function is nearly zero when the two points are close to each other. The structure function is also called the *variogram* in the Geostatistical literature [32]. A simple calculation shows that the structure function is related to the correlation function:

$$\mathbf{D_f}(\mathbf{x},\mathbf{x}') = \mathbf{C_f}(\mathbf{x},\mathbf{x}) + \mathbf{C_f}(\mathbf{x}',\mathbf{x}') - 2\mathbf{C_f}(\mathbf{x},\mathbf{x}'). \tag{3.3}$$

In the remainder of this section we will assume that the mean of the random function is constant and equal to the zero vector $\mathbf{0}$. This situation can always be achieved by considering the deviation $\mathbf{f}' = \mathbf{f} - \bar{\mathbf{f}}$ instead.

3.1.3 Spectral Representation of Random Functions

Homogeneity is the simplest possible level of order because it is the most elementary structural scheme that can be subjected to ordering. *Rudolf Arnheim*

The spectral representation of a function obtained via the Fourier transform is important in many applications. This representation characterizes the variations of a function and allows many operations such as filtering, to be performed or analyzed more efficiently. It is therefore natural to seek similar representations for the more general class of random functions. It should be noted that the Fourier transform (involving a Stieltjes integral) only exists for functions which are either periodic or which decay fast enough at infinity. Consequently we do not expect arbitrary random functions to have a spectral representation. The class of random functions which are the superposition of N harmonics with random amplitudes $\hat{\mathbf{f}}_j$

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} \hat{\mathbf{f}}_j \; e^{i\mathbf{k}_j^T\mathbf{x}},$$

clearly has a spectral representation, through superposition, which is given by the discrete set of amplitudes. In the case when the amplitudes are independent and have zero mean, the correlation function is given by [106]:

$$\mathbf{C}_{\mathbf{f}}(\mathbf{x},\mathbf{x}') = \sum_{j=1}^{N} \mathbf{S}_j \; e^{i\mathbf{k}_j^T(\mathbf{x}'-\mathbf{x})}, \tag{3.4}$$

where

$$\mathbf{S}_j = \langle|\hat{\mathbf{f}}_j|^2\rangle\mathbf{I}_m.$$

In particular, the correlation function only depends on the difference $\mathbf{r} = \mathbf{x}' - \mathbf{x}$ and has a discrete spectral representation given by the $\mathbf{S}_j$. Unfortunately, most random functions are not equal to a finite sum of harmonics. These results, however, can be generalized to the class of *homogeneous* random functions which share the property that their correlation function depends only on the difference $\mathbf{r} = \mathbf{x}' - \mathbf{x}$ between two points:

$$\mathbf{C}_{\mathbf{f}}(\mathbf{x},\mathbf{x}') \longleftarrow \mathbf{C}_{\mathbf{f}}(\mathbf{r}).$$

In particular, random functions which are the sum of a finite number of harmonics are homogeneous. The generalization is given by the *Wiener-Khinchin theorem*. This theorem states that the correlation function of a homogeneous random function satisfying

$$\int_{\mathbf{R}^d} \mathbf{C}_{\mathbf{f}}(\mathbf{r}) \; d\mathbf{r} < \infty$$

has a spectral representation given by a *spectral density function* $\mathbf{S}_{\mathbf{f}}(\mathbf{k})$:

$$\mathbf{C}_{\mathbf{f}}(\mathbf{r}) = \int_{\mathbf{R}^d} \mathbf{S}_{\mathbf{f}}(\mathbf{k}) \; e^{i\mathbf{k}^T\mathbf{r}} \; d\mathbf{k}.$$

Moreover, because the correlation is real valued, the spectral density satisfies

$$\mathbf{S}_{\mathbf{f}}(-\mathbf{k}) = \mathbf{S}_{\mathbf{f}}^*(\mathbf{k}).$$

In particular its diagonal elements are real values. The theorem also states that the diagonal elements are positive and was first proved for homogeneous random functions by Khinchin in 1933. However, in 1930, Wiener proved the result for deterministic functions and their correlation functions defined using spatial averages. For the class of ergodic random functions, Wiener's proof is therefore applicable, hence the name of the theorem[2]. A consequence of the theorem is that the trace of the spectral density

$$E_0(\mathbf{k}) = \frac{1}{2}\mathrm{tr}\mathbf{S}_{\mathbf{f}}(\mathbf{k}),$$

[2]Apparently Einstein was already aware of the result and proved a special case in 1914 [106]. Therefore, the theorem should actually be called the *Einstein-Wiener-Khinchin theorem*!

is a positive function which gives the contribution of each frequency $\mathbf{k}$ to the average total energy of the random function:

$$\langle |\mathbf{f}(\mathbf{x})|^2 \rangle = \mathrm{tr}\mathbf{C}_{\mathbf{f}}(\mathbf{0}) = \int_{\mathbf{R}^d} \mathrm{tr}\mathbf{S}_{\mathbf{f}}(\mathbf{k})\, d\mathbf{k},$$

The function E_0 is appropriately called the *energy spectrum* of the random function. An important consequence of the Wiener-Khichine theorem is that any homogeneous random function can be approximated arbitrarily closely (on average) by a finite sum of random harmonics. In the limit a spectral representation $\hat{\mathbf{f}}$ for the random function is obtained [41, 106]:

$$\mathbf{f}(\mathbf{x}) = \int_{\mathbf{R}^d} e^{i\mathbf{k}^T\mathbf{x}}\, d\hat{\mathbf{f}}(\mathbf{k}). \tag{3.5}$$

The random function $\hat{\mathbf{f}}$ is called the *Fourier transform* or the *spectral representation* of the random function $\mathbf{f}$. The spectral representation has the property that it is uncorrelated and related to the spectral density:

$$\langle d\hat{\mathbf{f}}(\mathbf{k}) d\hat{\mathbf{f}}^*(\mathbf{k}') \rangle = \delta(\mathbf{k}' - \mathbf{k}) \mathbf{S}_{\mathbf{f}}(\mathbf{k})\, d\mathbf{k}, \tag{3.6}$$

where δ is the delta function. The integral of Equation 3.5 is written as a Stieltjes integral. A Riemann integral in general does not exist since the spectral representation is nowhere differentiable. This is a somewhat surprising result which follows from Equation 3.6. Indeed for a small deviation $\Delta\mathbf{k}$ it implies that

$$\Delta\hat{\mathbf{f}}(\mathbf{k})^2 = \langle |\hat{\mathbf{f}}(\mathbf{k} + \Delta\mathbf{k}) - \hat{\mathbf{f}}(\mathbf{k})|^2 \rangle \propto \Delta\mathbf{k},$$

hence the limit of $\Delta\hat{\mathbf{f}}(\mathbf{k})/\Delta\mathbf{k} \propto 1/\sqrt{\Delta\mathbf{k}}$ diverges for $\Delta\mathbf{k}$ tending towards zero.

In some applications the assumption of homogeneity is too strong. Fluctuation of wind fields for example are homogeneous only over short periods of times, becoming strongly inhomogeneous over longer periods. Compare the speed of the wind during the day to that of the night for example. A less stringent condition is to assume that the structure function depends only on the difference between two points. Random functions satisfying this condition are called *locally homogeneous*. The condition depends on the differences between values of the random function, not on the values themselves and is therefore weaker than the condition of homogeneity. Indeed any homogeneous random function is also locally homogeneous. This follows directly from the simple relation (see Equation 3.3):

$$\mathbf{D}_{\mathbf{f}}(\mathbf{r}) = 2(\mathbf{C}_{\mathbf{f}}(\mathbf{0}) - \mathbf{C}_{\mathbf{f}}(\mathbf{r})). \tag{3.7}$$

The converse, however, is not true. A counter example is given by *random fractals* which have been used extensively in computer graphics. A random fractal, by definition is a scalar valued random function F having the following structure function:

$$D_F(\mathbf{r}) = |\mathbf{r}|^{2(d+1-D_f)},$$

where D_f is the *fractal dimension* of the random fractal [50]. The value of the fractal dimension lies between d and $d+1$ and characterizes the "roughness" of the random fractal. A result similar to the Wiener-Khinchin theorem can be derived for locally homogeneous random functions:

$$\mathbf{D}_{\mathbf{f}}(\mathbf{r}) = 2\int_{\mathbf{R}^d} \left(1 - e^{i\mathbf{k}\cdot\mathbf{r}}\right) \mathbf{S}_{\mathbf{f}}(\mathbf{k})\, d\mathbf{k}.$$

In fact this relation follows directly from Equation 3.7 when the random function is homogeneous. The surprising fact is that it remains true for non-homogeneous random functions. For example, the spectral density of a random fractal can be calculated using this theorem to be

$$S_F(\mathbf{k}) = |\mathbf{k}|^{-\beta},$$

where $\beta = 2d - 2D_f + 3$ [99].

3.1.4 Isotropic Incompressible Random Functions

From the point of view of the topologist or analyst, for whom the coninuum is a working reality, the existence of countable models means that formal language limitations as a means of imitating intuitive reasoning. [...] For the psychologist or philosopher, perhaps the most interesting aspect of the situation is that any mathematician can understand the viewpoint of another mathematician (without having to agree to it). This means that what mathematician A says, although demonstratively incapable of conveying unambiguous information about the continuum, nevertheless is capable of bringing the brain of mathematician B to a point where it forms an idea of the continuum which adequately represents the ideas of A's brain. Then B is still free to reject this idea. *Yu Manin*

In the previous section we mentioned an important class of homogeneous random functions. A further simplification is to assume that in addition, the correlation function is invariant under arbitrary rotations and reflections. These conditions constrain both the correlation function and the spectral density to a very particular form when the dimension d of the domain of the random function is equal to the dimension m of its values. In particular this is the case for two and three dimensional vector fields. Since we use this particular form to generate our turbulent motion fields we give an outline of the proof in order to make the result less mysterious. We give the derivation here in full vector notation, which clarifies the steps in the proofs. Such a proof was not found in the texts on which it is based. The derivation given here is based on the general theory of invariants and follows [80]. The first step in the derivation is to consider the bilinear form corresponding to the correlation function:

$$b(\mathbf{r}; \mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{C_f}(\mathbf{r}) \mathbf{b}, \tag{3.8}$$

where $\mathbf{a}$ and $\mathbf{b}$ are arbitrary vectors. A non-trivial theorem from the theory of invariants of the group of rotations and reflections states that the bilinear form can be expressed only in terms of the following fundamental invariants [104]:

$$\mathbf{a}^T\mathbf{b}, \quad \mathbf{a}^T\mathbf{r}, \quad \mathbf{r}^T\mathbf{b} \quad \text{and} \quad \mathbf{r}^T\mathbf{r} = r^2.$$

The only possible combination which yields a bilinear form is given by:

$$b(\mathbf{r}; \mathbf{a}, \mathbf{b}) = c_1(r)(\mathbf{a}^T\mathbf{b}) + c_2(r)(\mathbf{a}^T\mathbf{r})(\mathbf{r}^T\mathbf{b}),$$

Where c_1 and c_2 are two arbitrary scalar functions depending on the distance r only. Comparing with Equation 3.8, the correlation function is therefore given by

$$\mathbf{C_f}(\mathbf{r}) = c_1(r)\mathbf{I}_m + c_2(r)\mathbf{r}\mathbf{r}^T.$$

Similarly, the spectral density has to be of the form

$$\mathbf{S_f}(\mathbf{k}) = s_1(k)\mathbf{I}_m + s_2(k)\mathbf{k}\mathbf{k}^T, \tag{3.9}$$

with $k = |\mathbf{k}|$. These results imply that a stochastic model for an isotropic phenomenon is entirely specified by two one-dimensional scalar functions. This is true in any dimensions $d = m$. The assumption of isotropy when applicable is therefore very powerful.

The stochastic model can be reduced further if additional constraints are imposed on the random function. One such condition is to impose that the random function is *incompressible*:

$$\nabla^T \mathbf{f}(\mathbf{x}) = 0 \qquad \text{everywhere.} \tag{3.10}$$

This condition arises in many physical problems, most notably in the hydrodynamic equations of a fluid (see Section 2.3.2). We now show how this condition establishes a relation between the functions s_1 and s_2. To obtain an equation involving the spectral density we multiply Equation 3.10 by the random function evaluated at another location $\mathbf{x}+\mathbf{r}$ and average:

$$\langle \mathbf{f}(\mathbf{x}+\mathbf{r})\nabla^T \mathbf{f}(\mathbf{x})\rangle = \nabla^T \mathbf{C_f}(\mathbf{r}) = \mathbf{0}.$$

Differentiation in the spatial domain becomes a multiplication by a frequency in the spectral domain. Therefore

$$(i\mathbf{k})^* \mathbf{S_f}(\mathbf{k}) = \mathbf{0}.$$

We obtain an equation for the functions s_1 and s_2 by inserting Equation 3.9 into this relation:

$$(i\mathbf{k})^* s_1(k)\mathbf{I}_m + s_2(k)(i\mathbf{k})^*\mathbf{k}\mathbf{k}^T = (s_1(k) + k^2 s_2(k))(i\mathbf{k})^* = \mathbf{0}.$$

Because the frequency $\mathbf{k}$ is arbitrary we have $s_2(k) = -s_1(k)k^{-2}$. Therefore, the spectral density depends only on a single one dimensional scalar function s_1. We now give a physical meaning to this function by relating it to the energy spectrum of the random function. For isotropic random functions the energy spectrum E_0 is constant for all frequencies having the same length k. The consequent energy spectrum is thus a function of this length only. The contributions of all frequencies of length k to the total energy of the random function is given by the *isotropic energy spectrum* $E(k)$:

$$E(k) = \int_{|\mathbf{k}|=k} E_0(\mathbf{k})\, d\mathbf{k} = \frac{2\pi^{\frac{m}{2}}}{\Gamma(m/2)} k^{m-1} E_0(\mathbf{k}), \quad \text{where} \quad |\mathbf{k}| = k. \tag{3.11}$$

In other words $E(k)$ is equal to the energy spectrum E_0 multiplied by the surface area of the m-dimensional unit sphere[3]. Taking the trace of the expression for the spectral density given in Equation 3.9 we obtain:

$$E_0(\mathbf{k}) = \frac{1}{2}\mathrm{tr}\mathbf{S_f}(\mathbf{k}) = \frac{1}{2}s_1(k)\left(\mathrm{tr}\mathbf{I}_m - \frac{\mathrm{tr}(\mathbf{k}\mathbf{k}^T)}{k^2}\right) = \frac{1}{2}s_1(k)(m-1). \tag{3.12}$$

From Equations 3.9, 3.11 and 3.12 we obtain an expression for the spectral density in terms of the energy spectrum only:

$$\mathbf{S_f}(\mathbf{k}) = \left(\frac{E(k)\Gamma(m/2)}{\pi^{\frac{m}{2}}(m-1)k^{m-1}}\right)\left(\mathbf{I}_m - \frac{\mathbf{k}\mathbf{k}^T}{k^2}\right) = \Phi_m^2(k)\mathbf{P}(\mathbf{k}),$$

[3] Γ denotes the *gamma function*, see, e.g., [77] for its definition

where

$$\Phi_m(k) = \left(\frac{E(k)\Gamma(m/2)}{\pi^{\frac{m}{2}}(m-1)k^{m-1}} \right)^{\frac{1}{2}}.$$

The function $\mathbf{P}$ has the properties of a projector. First, it is idempotent:

$$\mathbf{P}^2(\mathbf{k}) = \left(\mathbf{I}_m - \frac{\mathbf{k}\mathbf{k}^T}{k^2} \right)^2 = \mathbf{I}_m - 2\frac{\mathbf{k}\mathbf{k}^T}{k^2} + \frac{\mathbf{k}(\mathbf{k}^T\mathbf{k})\mathbf{k}^T}{k^2 k^2} = \mathbf{P}(\mathbf{k}).$$

In fact this operator projects a vector onto the plane normal to the frequency $\mathbf{k}$. Second, we have that $\mathbf{P}^T(\mathbf{k}) = \mathbf{P}(\mathbf{k})$ and therefore the spectral density can be rewritten as:

$$\mathbf{S_f}(\mathbf{k}) = (\Phi_m(k)\mathbf{P}(\mathbf{k}))\,(\Phi_m(k)\mathbf{P}(\mathbf{k}))^T. \tag{3.13}$$

This is an important property of the spectral density and will be used in Section 3.4.3 to synthesize turbulent wind fields. In particular for $m = 3$, which corresponds to a random three-dimensional vector field, the spectral density is

$$\mathbf{S_f}(\mathbf{k}) = \frac{E(k)}{4\pi k^2} \left(\mathbf{I}_3 - \frac{\mathbf{k}\mathbf{k}^T}{k^2} \right). \tag{3.14}$$

These equations remain true for the spectral density density of *locally isotropic* random functions that have a structure function which is invariant under rotations and reflections. In particular, the structure function of a locally isotropic and incompressible random function must be of the following form [67]:

$$\mathbf{D_f}(\mathbf{r}) = \left(D_{lat}(r) + \frac{r}{2} D'_{lat}(r) \right) \mathbf{I}_m - \frac{1}{2r} D'_{lat}(r)\mathbf{r}\,\mathbf{r}^T,$$

where the *lateral structure* D_{lat} is related to the isotropic energy spectrum through the following "transform":

$$D_{lat}(r) = 4 \int_0^\infty \left(\frac{1}{3} - \frac{\sin kr}{(kr)^3} + \frac{\cos kr}{(kr)^2} \right) E(k)\, dk.$$

In Section 3.4.2 we will discuss different models for the the isotropic energy spectrum based on physical principles.

3.2 Transformations of Random Functions

3.2.1 Derivation of Stochastic Models

In many situations a phenomenon B is caused by another phenomenon A. For example, the wind creates waves and ripples on the surface of the ocean. In the case where a stochastic model is known for phenomenon A, a natural question to ask is whether one can derive a stochastic model for the resulting phenomenon B. For example, a stochastic model for the motions of ocean waves can be derived from a stochastic model of turbulent wind fields and the equations of hydrodynamics [41]. The advantage of this method is that a direct simulation to obtain a realization of B would not be necessary if it could be synthesized directly. This

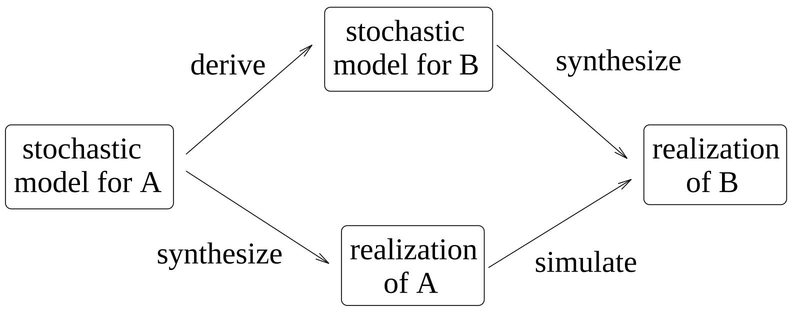

Figure 3.5: Derivation versus simulation.

difference is illustrated in Figure 3.5. More generally let $\mathbf{g}$ be a random function which models a phenomenon A causing a particular phenomenon B. The problem then becomes of finding a stochastic model for the random function $\mathbf{f}$ representing the phenomenon B. The relation can be formalized through a mapping $\mathcal{F}$:

$$\mathbf{f} = \mathcal{F}\mathbf{g}.$$

A second-order stochastic model for phenomenon B is then obtained by calculating the mean and the correlation function of the random function $\mathbf{f}$:

$$\begin{aligned} \bar{\mathbf{f}}(\mathbf{x}) &= \langle \mathcal{F}\mathbf{g}(\mathbf{x}) \rangle, \\ \mathbf{C}_{\mathbf{f}}(\mathbf{x}, \mathbf{x}') &= \langle \mathcal{F}\mathbf{g}(\mathbf{x})(\mathcal{F}\mathbf{g}(\mathbf{x}'))^T \rangle. \end{aligned} \tag{3.15}$$

For example we can calculate these functions for the linear operator of differentiation. The derivative of a scalar valued homogeneous random function g is given by

$$\mathbf{f}(\mathbf{x}) = \nabla g(\mathbf{x}).$$

It can be shown that the correlation function of the derivative is given by [106]:

$$\mathbf{C}_{\mathbf{f}}(\mathbf{r}) = -\nabla\nabla^T C_g(\mathbf{r}).$$

In other words the correlation function of the derivative of a random function is given by the second derivative of the correlation of the random function. In the spectral domain we have:

$$\mathbf{S}_{\mathbf{f}}(\mathbf{k}) = (i\mathbf{k})(i\mathbf{k})^* S_g(\mathbf{k}) = S_g(\mathbf{k})\mathbf{k}\mathbf{k}^T.$$

The result can be extended to the derivative of an arbitrary random function $\mathbf{g}$ by applying the above results to each of its components g_i.

The same procedure directly using Equation 3.15 can be carried out for the linear operator of integration applied to a one-dimensional random function $\mathbf{g}$:

$$\mathbf{f}(x) = \int_0^x \mathbf{g}(y)\, dy.$$

In this case we get the following relation:

$$\bar{\mathbf{f}}(x) = \int_0^x \bar{\mathbf{g}}(y)\, dy \quad \text{and} \quad \mathbf{C}_{\mathbf{f}}(x, x') = \int_0^x \int_0^{x'} \mathbf{C}_{\mathbf{g}}(y, y')\, dy\, dy'. \tag{3.16}$$

This result suggests an efficient algorithm to compute realizations of the integral of a random function. Instead of integrating realizations of the input random function, the realization of the integral can be computed directly from its mean and correlation function. The integrals of Equation 3.16 only have to be computed once. This idea of efficiently computing integrals of random functions is a key ingredient to the stochastic rendering of density fields, which will be presented in Section 5.6. Next we describe the application of Equation 3.15 to the filtering of random functions.

3.2.2 Quasi-Homogeneous Random Functions

A *quasi-homogeneous* random function $\mathbf{f}$ is defined as a position varying affine transformation of a homogeneous random function $\mathbf{g}$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{H}_1(\mathbf{x}) + \mathbf{H}_2(\mathbf{x})\mathbf{g}(\mathbf{x}). \tag{3.17}$$

The two functions $\mathbf{H}_1$ and $\mathbf{H}_2$ are deterministic. The mean of the quasi-homogeneous random function is equal to $\mathbf{H}_1$ and the correlation is equal to:

$$\mathbf{C}_{\mathbf{f}}(\mathbf{x}, \mathbf{x}') = \mathbf{H}_2(\mathbf{x})\mathbf{C}_{\mathbf{g}}(\mathbf{x}' - \mathbf{x})\mathbf{H}_2(\mathbf{x}')^T,$$

A realization of the random function can then be generated directly from a realization $\mathbf{g}_\omega$ of the homogeneous random function:

$$\mathbf{f}_\omega(\mathbf{x}) = \mathbf{H}_1(\mathbf{x}) + \mathbf{H}_2(\mathbf{x})\mathbf{g}_\omega(\mathbf{x}).$$

We will use this characterization to model the intensity field due to a random density function in Section 5.6.

3.2.3 Filtering of Random Functions

Filtering is a powerful technique to construct random functions. The results that are presented in this section will form the basis of the synthesis techniques of the next section. A filter linearly transforms an input random function $\mathbf{g}$ into an output random function $\mathbf{f}$ and is specified entirely by its *filter kernel* $\mathbf{H}$. More precisely

$$\mathbf{f}(\mathbf{x}) = \int_{\mathbf{R}^d} \mathbf{H}(\mathbf{x}, \mathbf{y})\mathbf{g}(\mathbf{y}) \, d\mathbf{y}. \tag{3.18}$$

The characteristics of the output random function are therefore determined completely by the form of the filter kernel and by a stochastic model of the input random function. This is the fact which lies at the heart of most synthesis techniques. First a random function $\mathbf{g}$ whose realizations can be computed efficiently is chosen. A suitable filter kernel is then selected such that the filtered random function has a desired stochastic model. Equation 3.15 for the linear filtering operator becomes:

$$\bar{\mathbf{f}}(\mathbf{x}) = \int_{\mathbf{R}^d} \mathbf{H}(\mathbf{x}, \mathbf{y})\bar{\mathbf{g}}(\mathbf{y}) \, d\mathbf{y},$$
$$\mathbf{C}_{\mathbf{f}}(\mathbf{x}, \mathbf{x}') = \int_{\mathbf{R}^d} \int_{\mathbf{R}^d} \mathbf{H}(\mathbf{x}, \mathbf{y})\mathbf{C}_{\mathbf{g}}(\mathbf{y}, \mathbf{y}')\mathbf{H}^T(\mathbf{x}', \mathbf{y}') \, d\mathbf{y}' \, d\mathbf{y}. \tag{3.19}$$

In principle then, the filter kernel $\mathbf{H}$ can be chosen such that a random function with a desired correlation function $\mathbf{C_f}$ is obtained. The equation, however, is very hard to solve for the kernel in general. A more tractable solution can be obtained for homogeneous random functions under shift-invariant filtering. In this case,

$$\mathbf{f}(\mathbf{x}) = \int_{\mathbf{R}^d} \mathbf{H}(\mathbf{x}-\mathbf{y})\mathbf{g}(\mathbf{y})\, d\mathbf{y}.$$

In particular the filter kernel is less complicated when the dimension of its domain is halved. By definition the output random function is also homogeneous, and consequently both the input and the output random functions have a spectral representation. If we denote the Fourier transform of the kernel by $\hat{\mathbf{H}}$, then shift invariant filtering becomes a simple multiplication in the spectral domain:

$$d\hat{\mathbf{f}}(\mathbf{k}) = \hat{\mathbf{H}}(\mathbf{k})d\hat{\mathbf{g}}(\mathbf{k}). \tag{3.20}$$

This fact is of course crucial to signal theory. The relation between the spectral density of the input and the spectral density of the resulting random function is given by:

$$\mathbf{S_f}(\mathbf{k}) = \hat{\mathbf{H}}(\mathbf{k})\mathbf{S_g}(\mathbf{k})\hat{\mathbf{H}}^*(\mathbf{k}). \tag{3.21}$$

This relation in general is easy to solve for the transform of the filter kernel, and it is the key equation for our spectral synthesis technique described in Section 3.3.2.

3.3 Synthesis Algorithms

In this section we present several synthesis algorithms to generate realizations of a random function from its stochastic model. The synthesis techniques are presented at a high level of generality and potentially could be applied to a wide range of problems. To our knowledge, these algorithms have never before appeared at this level of generality. In this way, they constitute an important contribution of this thesis to the field. Later in this thesis these techniques will be used to generate turbulent wind fields (Section 3.4.3) and random transparency fields (Section 5.6). All synthesis methods presented in this section use only the mean and the correlation function to generate samples of a random function. For homogeneous random functions we describe an efficient synthesis technique based on the spectral density. One important random function is *white noise* from which many other random functions can be generated. This random function is described next.

3.3.1 White Noise

A white noise random function is characterized by a total lack of coherence, representing therefore a state of absolute chaos. More precisely a white noise $\mathbf{w}$ has a zero mean and a correlation function equal to:

$$\mathbf{C_w}(\mathbf{r}) = \delta(\mathbf{r})\mathbf{I}_m,$$

where δ is the delta distribution. Strictly speaking white noise is not an ordinary random function, because its correlation is a distribution. The theory of random functions, however, can be extended to the more general class of distributions [41]. For example, the spectral density of a white noise can be calculated through the generalized Wiener-Khinchin theorem and the sifting property of the δ to be:

$$\mathbf{S}_{\mathbf{w}}(\mathbf{k}) = \mathbf{I}_m.$$

The properties of a white noise are therefore entirely defined by its one point probability distribution. In the case when this distribution is Gaussian one speaks of *Gaussian white noise*. Before discussing algorithms to generate white noise, we discuss algorithms to synthesize a realization of a random variable.

Synthesizing Random Variables

Let us assume that a random variable is scalar valued. Its stochastic model is therefore entirely given by a single valued probability distribution. In the case that this distribution is uniform on an interval $[0, 1]$, realizations can be synthesized directly using a pseudo-random number generator. These generators are based on simple iterative schemes with a high period. They can be found on most systems, e.g., `drand48()` on UNIX platforms. There are many techniques to generate realizations of random variables having an arbitrary distribution function $F(z)$ from a realization of the uniform distribution. The *inversion method* permits realizations of random variables to be calculated from realizations of uniform random variables. This method can be used only if the distribution function F is invertible. The method is quite simple: first generate a realization X_ω of a uniformly distributed random variable, then compute $Y_\omega = F^{-1}(X_\omega)$. However, the Gaussian distribution function cannot be inverted easily and the inversion method cannot be applied. An algorithm to generate a realization of a Gaussian random variable is based on the central limit theorem. First generate N uncorrelated realizations of a uniformly distributed random variable X. The sum $S_\omega = X_{1,\omega 1} + \cdots + X_{N,\omega}$ of these realizations is then approximately a realization of a Gaussian random variable with mean $N/2$ and variance $N/12$. A Gaussian random variable with zero mean and unit variance is obtained by centring and scaling the sum [106]:

$$S'_\omega = \frac{S_\omega - N/2}{\sqrt{N/12}} = \sqrt{\frac{12}{N}} \left(S_\omega - \frac{1}{2} \right).$$

Good results are obtained for values as small as $N = 8$. We now discuss several algorithms to approximate a white noise.

Spatial Methods

In spatial domain, a white noise can be approximated by a *Poisson noise process*. The Poisson process is given by a sum of uncorrelated random variables $\mathbf{v}_j$ distributed at uncorrelated random locations $\mathbf{x}_j$ [106]:

$$\mathbf{w}_\omega(\mathbf{x}) \approx \sum_{j=1}^{N} \mathbf{v}_{j,\omega} \delta(\mathbf{x} - \mathbf{x}_{j,\omega}). \tag{3.22}$$

The realizations of the random variables $\mathbf{v}_j$ are generated using either the uniform or the Gaussian random number generators discussed in the previous subsection. An efficient scheme based on lookup tables is described in [72]. Realizations for the random locations can be generated by first subdividing the space into a grid. A fixed number of realizations are then generated within each grid-cell [49]. A white noise can also be approximated by a set of uncorrelated random variables $\mathbf{v}_j$ defined at a fixed number of sample locations $\mathbf{x}_j$. This is a special case of Poisson noise where the locations are deterministic. Values of the white noise at any point are then approximated through an interpolant Φ:

$$\mathbf{w}_\omega(\mathbf{x}) \approx \sum_{j=1}^{N} \mathbf{v}_{j,\omega}\Phi(\mathbf{x}-\mathbf{x}_j).$$

For samples defined on a regular grid many choices for the interpolant Φ are possible, including polynomial splines [72].

Spectral Methods

In the spectral domain an approximation of a realization of a white noise can be obtained by assigning N independent uncorrelated complex random variables $\mathbf{v}_j$ to a set of discrete wave numbers $\mathbf{k}_j$. From the spectral representation (see Eq. 3.5) a realization for the white noise is approximated by:

$$\mathbf{w}_\omega(\mathbf{x}) \approx \sum_{j=1}^{N} \mathbf{v}_{j,\omega} e^{i\mathbf{k}_{j,\omega}^T\mathbf{x}}, \tag{3.23}$$

i.e., by a finite number of random "waves". A realization for each of the components of the random variables $\mathbf{v}_j$ is computed by generating a Gaussian distributed amplitude A and a uniformly distributed phase ψ and equating it to $Ae^{i2\pi\psi}$.

3.3.2 Spectral Synthesis

We now describe a synthesis technique for homogeneous random functions. Any homogeneous random function $\mathbf{f}$ can be represented as the convolution of a white noise with a kernel $\mathbf{H}$. From Equation 3.21 the kernel is defined by

$$\mathbf{S_f}(\mathbf{k}) = \hat{\mathbf{H}}(\mathbf{k})\mathbf{S_w}(\mathbf{k})\hat{\mathbf{H}}^*(\mathbf{k}) = \hat{\mathbf{H}}(\mathbf{k})\hat{\mathbf{H}}^*(\mathbf{k}).$$

Because the spectral density is a positive definite symmetric matrix, the matrix $\hat{\mathbf{H}}$ is actually real:

$$\mathbf{S_f}(\mathbf{k}) = \hat{\mathbf{H}}(\mathbf{k})\hat{\mathbf{H}}^T(\mathbf{k}). \tag{3.24}$$

There is an infinite set of kernels that have this property: the kernel $\mathbf{H}' = \hat{\mathbf{H}}\mathbf{Q}'$ with $\mathbf{Q}'$ an orthonormal transformation also satisfies Eq. 3.24. A unique decomposition is obtained when an additional $m(m-1)/2$ constraints are added. For example, there is a stable algorithm called the *Cholesky Decomposition* which calculates $\hat{\mathbf{H}}$ from the spectral density such that all its elements below the diagonal are zero. The complexity of this decomposition is $O(m^3)$. Using the approximation of a white noise given in Equation 3.23 and the relationship between the random spectral representations given in Eq. 3.20, it is possible to approximate a realization of the random function $\mathbf{f}$ by:

$$\mathbf{f}_\omega(\mathbf{x}) \approx \sum_{j=1}^{N} \hat{\mathbf{H}}(\mathbf{k}_j)\mathbf{v}_{j,\omega}\, e^{i\mathbf{k}_j^T\mathbf{x}} = \sum_{j=1}^{N} \hat{\mathbf{f}}_j\, e^{i\mathbf{k}_j^T\mathbf{x}}. \tag{3.25}$$

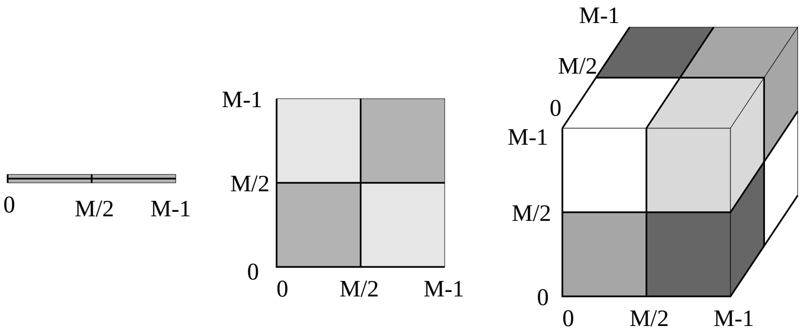

Figure 3.6: Symmetry relation for the fast Fourier transform in one, two and three dimensions.

We now describe an algorithm based on the Fast Fourier Transform (FFT) when the frequencies are sampled on a regular grid of dimension $N = M^d$. We label these frequencies using a multi-index $\mathbf{a} \in \{0, \cdots, M-1\}^d$, i.e., $\mathbf{k}_j \rightarrow \mathbf{k}_{\mathbf{a}}$. In order for the realization of the random functions to be real valued, the sampled frequencies have to satisfy the following symmetries [77]:

$$\hat{\mathbf{f}}_{\mathbf{a}} = \hat{\mathbf{f}}_{(a_1,a_2,\cdots,a_d)} = \hat{\mathbf{f}}^*_{(M-a_1,M-a_2,\cdots,M-a_d)} = \hat{\mathbf{f}}^*_{M\mathbf{1}-\mathbf{a}} \quad \forall \mathbf{a} \in \{0, \cdots, M-1\}^d,$$

where $\mathbf{1} = (1, 1, \cdots, 1)$ and the indices are taken modulo M such that $M - 0 = 0$. In Figure 3.6 we illustrate the geometric meaning of these symmetries for the cases $d = 1, 2, 3$. In particular, the symmetry conditions imply that elements whose index satisfies $\mathbf{a} = M\mathbf{1} - \mathbf{a}$ have a zero imaginary part. These are exactly the multi-indices whose elements are either 0 or $M/2$. The symmetry conditions imply that only half of the terms have to be computed. The following algorithm generates all coefficients with the desired properties:

```
for $\mathbf{b} \in \{0, \cdots, M-1\}^{d-1}$ do
    for $k \in \{0, \cdots, M/2\}$ do
        $\mathbf{a} = (\mathbf{b}, k)$
        *solve* Equation 3.24 for the kernel $\hat{\mathbf{H}}(\mathbf{k}_{\mathbf{a}})$
        *synthesize* a realization $\mathbf{v}_{\mathbf{a},\omega}$
        *compute* $\hat{\mathbf{f}}_{\mathbf{a}} = \hat{\mathbf{H}}(\mathbf{k}_{\mathbf{a}})\mathbf{v}_{\mathbf{a},\omega}$
        $\hat{\mathbf{f}}_{M\mathbf{1}-\mathbf{a}} = \hat{\mathbf{f}}^*_{\mathbf{a}}$
    end for
end for
for $\mathbf{a} \in \{0, M/2\}^d$ do
    set the imaginary part of $\hat{\mathbf{f}}_{\mathbf{a}}$ to zero
end for
```

An approximation of a realization of the random function is then obtained by taking m inverse FFTs, one for each component:

$$f_1 = \text{FFT}^{-1}(\hat{f}_1), \cdots, f_m = \text{FFT}^{-1}(\hat{f}_m).$$

The above algorithm can be generalized somewhat by having different numbers M_k of samples for each of the dimensions and the multi-index $\mathbf{a}$ then takes values in the space $\{0, \cdots, M_1 -$

$1\} \times \cdots \times \{0, \cdots, M_d - 1\}$. For example, for random functions evolving over time, a higher sampling rate for the time dimension might be necessary to remove periodic artifacts. The complexity of this algorithm is therefore the sum of the complexity of the initialization pass and the complexity of the m FFT's, and is therefore $O(m^3 N + mN \log N)$. The resulting field is defined in the spatial domain on a discrete set of samples $\mathbf{x_a}$. The amount of storage required is therefore $O(mN)$. The property of the set is that the random function is periodic on it. This can be either an advantage or a disadvantage, depending on the application. An advantage is that even a small set of samples defines a field which is defined everywhere in space. When the random function is directly observable, however, the periodicity might appear artificial which is a decided disadvantage. This method is not suited to problems in which the function is evaluated only at sparsely located points in space, because all the spatial samples have to be computed at once.

A similar algorithm was proposed by Shinozuka [87, 88]. As in our algorithm the a realization of a random function with prescribed spectral density is generated by filtering a white noise in the frequency domain. However, they did not implement the algorithm using the fast Fourier transform, but expressed the realization as a sum of weighted cosine functions.

In computer graphics, this synthesis technique was used by Voss to create fractal mountains and clouds [99], by Mastin et al. to generate random ocean waves [51] and by Sakas to generate random density distribution for gaseous phenomena [85]. These techniques were, however, restricted to scalar valued random functions. We have presented a generalization of these methods to arbitrary dimensions.

3.3.3 Spatial Synthesis

For many applications the spectral synthesis technique is not appropriate. Its main disadvantages are that the whole set of spatial samples has to be generated at once and that for some applications the periodicity is undesirable. It is also limited to locally homogeneous random functions. In this section we present some synthesis techniques which operate in the spatial domain. We start with a technique that works for random functions with arbitrary correlation functions and then give some alternatives to the spectral synthesis algorithms for homogeneous random functions.

Realizations of a non-homogeneous random function with arbitrary correlation $\mathbf{C_f}(\mathbf{x}, \mathbf{x}')$ can be generated by filtering white noise in the spatial domain. The filter in this case is determined by using Equation 3.19:

$$\mathbf{C_f}(\mathbf{x}, \mathbf{x}') = \int_{\mathbf{R}^n} \mathbf{H}(\mathbf{x}, \mathbf{y}) \mathbf{H}^T(\mathbf{x}', \mathbf{y})\, d\mathbf{y}.$$

In general, this equation cannot be solved analytically for the filter kernel. By discretizing the domain of the correlation function into N discrete points $\mathbf{x}_i$ an approximation $\mathbf{H}_{jk} = \mathbf{H}(\mathbf{x}_j, \mathbf{x}_k)$ can be obtained by numerically solving the linear system:

$$\mathbf{C}(\mathbf{x}_j, \mathbf{x}_k) = \sum_{l=1}^{N} \mathbf{H}_{jl} \mathbf{H}_{lk}^T. \qquad i, j = 1, \cdots, N.$$

Because the correlation function is positive definite and symmetric, this system of mN equations can be solved by a Cholesky decomposition with time complexity $O((mN)^3)$.

Once a solution is obtained for the discrete kernel $\mathbf{H}_{jk}$, a realization of the random function can be obtained on the discretization (see Eq. 3.18):

$$\mathbf{f}_\omega(\mathbf{x}_j) \approx \sum_{k=1}^{N} \mathbf{H}_{jk}\mathbf{v}_{k,\omega},$$

where the $\mathbf{v}_k$ are independent random variables synthesized using any one of the techniques described in Section 3.3.1. An approximation of a realization of the function can then be obtained in time $O((mN)^2)$. This method is therefore significantly more expensive than the spectral synthesis algorithm. However, this spatial synthesis algorithm is more general since it is not restricted to the class of homogeneous random functions.

We now present some techniques to generate homogeneous random functions in the spatial domain. The filter kernel is obtained as the inverse Fourier transform of the kernel $\hat{\mathbf{H}}$ obtained by solving Equation 3.21 analytically. In some instances the kernel is $\mathbf{H}$ is known in advance. A realization of the random function can be computed by filtering a Poisson noise approximation of a white noise. Inserting the Poisson noise defined by Equation 3.22 into Equation 3.18 we get the following approximation:

$$\mathbf{f}_\omega(\mathbf{x}) \approx \sum_{j=1}^{N} \mathbf{H}(\mathbf{x} - \mathbf{x}_{j,\omega})\mathbf{v}_{j,\omega}.$$

Lewis called this method *sparse convolution* and used it to synthesize scalar valued random functions [49]. For arbitrary filter kernels, the complexity of this algorithm is $O(m^2 N n_e)$, where n_e is the number of points at which the realization is evaluated. In cases where the filter kernel has a small support, the sum in the approximation becomes independent of N and the complexity therefore is dominated by the number of evaluations: $O(m^2 n_e)$. The storage requirements are also significantly lower: $O(m^2)$. This technique is therefore an improvement over the spectral synthesis technique when periodicity is not desired or when the random function has to be evaluated on a small sparse set of samples.

3.4 Turbulent Wind Fields

We now turn to an important application of the theory of random function. In this section we present a new stochastic model for evolving three-dimensional turbulent wind fields. This model is a key component in almost all of our simulations. Without a turbulent wind field, the animations would lack the chaotic look so characteristic of most natural phenomena. We first show how a turbulent wind field is a special case of a random function. Thereafter we give several models for the energy spectrum which entirely characterize isotropic incompressible turbulent wind fields. Finally we specialize the general spectral synthesis technique to the case of turbulent wind fields.

3.4.1 Stochastic Model

A *turbulent wind field* is a random function whose realizations are three-dimensional vector fields defined over space and time:

$$\mathbf{u}(\mathbf{x},t) = \begin{pmatrix} u_1(x_1,x_2,x_3,t) \\ u_2(x_1,x_2,x_3,t) \\ u_3(x_1,x_2,x_3,t) \end{pmatrix}.$$

We will assume that the turbulent wind field is homogeneous and locally isotropic in its spatial variable $\mathbf{x}$. This situation is highly idealized and is rarely encountered in reality. In fact only wind fields defined in a domain without boundaries can be exactly isotropic. Any boundary, such as a wall will introduce anisotropies into the wind field. The assumption of local isotropy implies that isotropy holds approximately only for the small structures of the wind field. Close approximations to locally isotropic wind fields are obtained in laboratory experiments by blowing air through a fine grid. However, in practice many real flows have spectral densities which are very close to the ones predicted by the isotropic theory. The results presented in this thesis are valid only for locally isotropic turbulent wind fields. Since we assume that the wind field is locally isotropic only in its spatial variable, it is convenient to separate each frequency into a *wave number* $\mathbf{k}$ and a *temporal frequency* κ. Hence, the formulas derived for locally isotropic random functions apply to the spatial variable only. In particular the energy spectrum[4] of the wind field can be written in terms of the temporal variable t or in terms of the spatial frequency κ. In the latter we will denote the energy spectrum by E'. For turbulent wind fields, the energy spectrum now has a clear physical meaning since it gives the contribution of all wave numbers of length k to the average kinetic energy of the wind field at time t:

$$\int_0^\infty E(k,t)\, d\mathbf{k} = \frac{1}{2}\langle|\mathbf{u}(\mathbf{x},t)|^2\rangle.$$

Following the derivations of Section 3.1.4, and in particular Equation 3.14, the spectral density is a function of the spectral density E only:

$$\mathbf{S_f}(\mathbf{k},t) = \frac{E(k,t)}{4\pi k^4}\left(k^2\mathbf{I}_3 - \mathbf{k}\mathbf{k}^T\right).$$

The goal of the statistical theory of locally isotropic turbulence is to develop models for the energy spectrum from various physical arguments. Some of the methodology and some of the models are presented in the next section.

3.4.2 The Energy Spectrum

The first step in characterizing locally isotropic turbulence is to derive an equation for the energy spectrum from the equations of Navier-Stokes. Such an equation can indeed be obtained by the following procedure [62]. Let $\mathrm{NS}(\mathbf{u}(\mathbf{x},t)) = 0$ denote the Navier-Stokes equation for the wind field. Then another equation can be constructed from it as follows:

$$\langle\mathbf{u}(\mathbf{x}+\mathbf{r},t)\mathrm{NS}(\mathbf{u}(\mathbf{x},t))^T + \mathrm{NS}(\mathbf{u}(\mathbf{x}+\mathbf{r},t))\mathbf{u}^T(\mathbf{x},t)\rangle = 0.$$

[4]We drop the "isotropic" qualifier since only locally isotropic random functions are considered in this section.

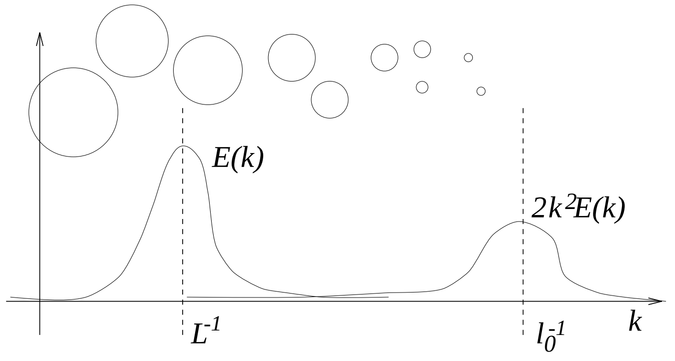

Figure 3.7: Energy spectrum versus kinetic energy transfer and corresponding eddies.

This equation establishes a relation between the correlation tensor and a tensor of order three. This relation is known as the *Karman-Howarth equations.* In the frequency domain this equation yields the following equation for the energy spectrum:

$$\left(\frac{\partial}{\partial t} + 2\nu k^2\right) E(k,t) = T(k,t), \tag{3.26}$$

where ν is the viscosity of the wind field. The function T corresponds to the triple velocity correlations coming from the non-linear interactions of the Navier-Stokes equations [47]. Since neither of the terms on the left hand side of the equation are responsible for the energy transfer between the different wave numbers, this function must be. Therefore it is called the *kinetic energy transfer* and can be interpreted as the transfer of kinetic energy through *eddies* of size k^{-1}. An "eddie" is loosely speaking, a visible structure of a certain size in the fluid. In fact an eddie is not localized in any way but rather refers to the set of structures having a given size. We now present the famous *Kolmogorov energy spectrum* following Chorin's derivation [12]. In the case where the the viscosity ν of the wind is very low, the flow is extremely turbulent and approaches a steady state, i.e.,

$$\frac{\partial E(k,t)}{\partial t} \to 0 \quad \text{and} \quad E(k,t) \to E(k).$$

From Equation 3.26 it then follows that the kinetic energy transfer is equal to

$$T(k) = 2\nu k^2 E(k).$$

In particular, the support of the energy spectrum and the kinetic energy transfer are disjoint as illustrated in Figure 3.7 because of the multiplication by k^2. The energy spectrum peaks at an eddie of size L and corresponds to the scales where the energy is injected into the wind field (by waving your hand or blowing for example). The kinetic energy transfer, on the other hand, peaks at smaller eddies of size l_0 near the dissipation range of the wind field. The eddies between these extremes are the *inertial range* in which the transfer from the larger eddies to the smaller eddies occurs. Kolmogorov postulated that the energy spectrum in the inertial range depends only on the length of the wave number and on the rate of dissipation of kinetic energy:

$$\epsilon = \frac{\partial \langle \mathbf{u}^2 \rangle}{\partial t}.$$

The sole combination of these quantities which has the units of energy is

$$E_K(k) = C_K \epsilon^{2/3} k^{-5/3} \quad \text{for} \quad 1/L < k < 1/l_0,$$

where C_K is a dimensionless constant which can be determined experimentally and is usually on the order of 1.5 [47]. The intuitive picture behind the Kolmogorov spectrum is that of an energy cascade from the larger eddies towards the smaller eddies (again see Figure 3.7). We will not present other energy spectra $E(k)$ for the steady state. These can be found for example in [62]. We will now present a simple method of including the time dependence into the energy spectrum.

A possible method of including a time dependence is to assume that the turbulence "travels" at a mean velocity $\mathbf{v} = (v_1, v_2, v_3)$:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(x_1 - t/v_1, x_2 - t/v_2, x_3 - t/v_3, 0).$$

This is known as the *Taylor Hypothesis* in the statistical fluid dynamics literature. This assumption has been used by Shinya and Fournier to simulate two-dimensional turbulent wind fields [89]. In this thesis we model the temporal frequency dependence of the energy spectrum $E'(k, \kappa)$ by multiplying the Kolmogorov energy spectrum $E_K(k)$ by a temporal spread function $G_k(\kappa)$:

$$E'(k, \kappa) = E_K(k) G_k(\kappa). \tag{3.27}$$

The temporal spread function is subject to:

$$\int_{\mathbf{R}} E'(k, \kappa)\, d\kappa = E_K(k) \int_{\mathbf{R}} G_k(\kappa)\, d\kappa = E_K(k).$$

This guarantees conservation of kinetic energy. 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 currents in 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:

$$G_k(\kappa) = \frac{1}{\sqrt{2\pi}\, k\sigma} \exp\left(-\frac{\kappa^2}{2k^2\sigma^2}\right). \tag{3.28}$$

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

3.4.3 Synthesis of Turbulent Wind Fields

To synthesize realizations of turbulent wind fields, we will use the spectral synthesis algorithm described in Section 3.3.2. The properties of the resulting field are particularly nice in the case of turbulent wind fields. The periodicity of the wind field on the grid allows us to have a field defined for all locations of space and all times. Even a small field, such as $M = 16$, has a structure which is rich enough to generate complicated motion. The periodicity is not noticeable because instead of observing the wind field directly we observe the effect of the wind field on other systems. The temporal evolution is very important in this context.

The effects of a periodic wind field in a steady state, i.e., frozen in time, become visible. Furthermore, because the values of the wind field are precomputed on a regular grid, it can be computed at all other locations very efficiently through interpolation. This is very important in the design process, because an animator can view the effect of the wind field in real time. For these reasons we have chosen the spectral synthesis technique to generate realizations of a turbulent wind field. This technique from Section 3.3.2 is repeated here for the special case of turbulent wind fields because of certain peculiarities.

We proved in Section 3.1.4 that the spectral density of an isotropic random function can be decomposed into a product of a projector and an energy spectrum (see Equation 3.13). The filter kernel producing such a spectral density from a white noise is then given by:

$$\hat{\mathbf{H}}(\mathbf{k},\kappa) = \Phi_3(k,\kappa)\mathbf{P}(\mathbf{k}),$$

where

$$\Phi_3(k,\kappa) = \left(\frac{E_K(k)G_k(\kappa)}{4\pi k^2}\right)^{\frac{1}{2}}.$$

We assume that the the wind field is sampled on a regular grid of size $M^3 \times M_t$, i.e., that the spatial resolution is given by M and that the temporal resolution is given by M_t. A realization for each sample $\hat{\mathbf{u}}_{i,j,k,l}$ is then obtained by first generating three uncorrelated realizations of a complex Gaussian random variable and then multiplying them by the transform of the filter kernel:

```
Compute $\hat{\mathbf{u}}_{i,j,k,l}$:
    *generate* three Gaussian random variables $A_1$, $A_2$ and $A_3$ and 3 uniformly distributed
            random variables $\psi_1$, $\psi_2$ and $\psi_3$.
    $v_1 = A_1 e^{i2\pi\psi_1}$, $v_2 = A_2 e^{i2\pi\psi_2}$ and $v_3 = A_3 e^{i2\pi\psi_3}$
    $k_1 = 2i/M$, $k_2 = 2j/M$, $k_3 = 2k/M$ and $\kappa = 2l/M_t$
    $k = \sqrt{k_1^2 + k_2^2 + k_3^2}$
    $(\hat{u}_{i,j,k,l})_1 = \Phi_3(k,\kappa)\left(\left(1-\frac{k_1^2}{k^2}\right)v_1 - \frac{k_1k_2}{k^2}v_2 - \frac{k_1k_3}{k^2}v_3\right)$
    $(\hat{u}_{i,j,k,l})_2 = \Phi_3(k,\kappa)\left(-\frac{k_2k_1}{k^2}v_1 + \left(1-\frac{k_2^2}{k^2}\right)v_2 - \frac{k_2k_3}{k^2}v_3\right)$
    $(\hat{u}_{i,j,k,l})_3 = \Phi_3(k,\kappa)\left(-\frac{k_3k_1}{k^2}v_1 - \frac{k_3k_2}{k^2}v_2 + \left(1-\frac{k_3^2}{k^2}\right)v_3\right)$
```

Given this definition all samples are initialized by the following algorithm which is a particular case of the initialization step given in Section 3.3.2. We assume that the spatial resolution is given by M and that the temporal resolution is given by M_t.

```
Initialize:
    for $i,j,k \in \{0,\cdots,M-1\}$ do
        for $l = 0,\cdots,M_t/2$ do
            Compute $\hat{\mathbf{u}}_{i,j,k,l}$
            $\hat{\mathbf{u}}_{M-i,M-j,M-k,M_t-l} = \hat{\mathbf{u}}^*_{i,j,k,l}$
        end for
    end for
    for $i,j,k \in \{0,M/2\}$ do
        Set imaginary part of $\hat{\mathbf{u}}_{i,j,k,0}$ and $\hat{\mathbf{u}}_{i,j,k,M_t/2}$ to zero
    end for
```

After the initialization step we perform three inverse FFT's to obtain a realization of the turbulent wind field. In Chapter 6 we show how this wind field is used in the depiction of smoke, steam and fire. To conclude this chapter we mention several other techniques to synthesize turbulent wind fields.

3.4.4 Other Synthesis Techniques

Random Frequencies

Several techniques have been developed which are based on models in which the spatial and temporal frequencies are chosen randomly. In this case the turbulent wind field is approximated by:

$$\mathbf{u}(\mathbf{x}, t) \approx \sum_{j=1}^{N} \mathbf{P}(\mathbf{k}_j)\mathbf{v}_j e^{i\mathbf{k}_j^T \mathbf{x} + \kappa_j t},$$

where the frequencies and the *real* amplitudes $\mathbf{v}_j$ are mutually independent random variables. If the random frequencies have an isotropic probability density function given by

$$\varphi(|\mathbf{k}|, \kappa) = \frac{E'(|\mathbf{k}|, \kappa)}{\langle |\mathbf{u}|^2 \rangle},$$

then the wind field has the required energy spectral density [40]. A synthesis algorithm is then obtained by first generating N independent frequencies. Each frequency is generated by the following algorithm:

```
Generate a sample $(k_\omega, \kappa_\omega)$ from the distribution $\varphi$.
Generate a uniformly distributed unit random vector $\hat{\mathbf{k}}_\omega$.
return $k_\omega \hat{\mathbf{k}}_\omega$
```

The amplitudes are generated from a Gaussian distribution. This synthesis technique is preferable if only a few random frequencies are sufficient.

Sparse Convolution

We can synthesize turbulent wind fields directly in the spatial domain using the sparse convolution technique presented in Section 3.3.3. For this method we need the convolution filter in the spatial domain. This can be achieved by taking the inverse Fourier transform of the filter kernel:

$$\hat{\mathbf{H}}(\mathbf{k}, \kappa) = \Phi_3(k, \kappa)\mathbf{P}(\mathbf{k}) = \Phi_3(k, \kappa)\left(I_3 - \frac{\mathbf{k}\mathbf{k}^T}{k^2}\right).$$

Once the Fourier transform has been calculated the field can be evaluated anywhere in space by generating a set of N independent realizations of random space-time locations $(\mathbf{x}_j, t_j)$ and a set of N realizations of a random vector $\mathbf{v}_j$. A realization of the wind field in the spatial domain is then given by:

$$\mathbf{u}_\omega(\mathbf{x}, t) = \sum_{j=1}^{N} \mathbf{H}(\mathbf{x} - \mathbf{x}_{j,\omega}, t - t_{j,\omega})\mathbf{v}_{j,\omega}.$$

In particular this method can be used to generate nonhomogeneous turbulent random functions by modifying the filter kernel locally. For example this can be achieved by multiplying the the smoothing kernel by a function f which depends on location and on time:

$$\mathbf{H}(\mathbf{x}-\mathbf{x}_{j,\omega}, t-t_{j,\omega}) \longrightarrow f(\mathbf{x},t)\mathbf{H}(\mathbf{x}-\mathbf{x}_{j,\omega}, t-t_{j,\omega}).$$

Such extensions have not been explored in this thesis but are a promising direction of future research.

Chapter 4

Solution of Equations on Unordered Data

The physical models used in this thesis for the propagation of light and the evolution of gases are partial differential equations (PDEs) defined in a three-dimensional space (see Chapter 2). The approximate solution of PDEs on three-dimensional grids is generally intractable. Indeed, let us assume that we want to sample a physical quantity having spatial structures with sizes ranging from a small distance l to the the size of the system L. In this case we need approximately $N = (L/l)^3$ samples on a regular grid. For example, a simulation of the dynamics of clouds involves spatial structures ranging from a couple of centimeters ($l = 0.01$) to a couple of kilometers ($L = 1000$). Such simulations therefore require that the physical properties of the cloud be stored at

$$(1000/0.01)^3 = (10,000)^3 = 10^{15}$$

grid points. Assuming that each property can be stored using only a couple of bytes, we would need a storage capacity of 10^9 megabytes! This result remains true even for simple advection-diffusion equations. Clearly then, in order to obtain tractable solutions, we have to develop ad hoc numerical methods for the particular system of equations under study. In this chapter we present a general method of solution for PDEs on an unordered set of samples. In particular an advection term in the PDE can be resolved by moving the samples along the advecting force. The data points in this case can thus be interpreted as "particles". However, these particles should not be mistaken for the elementary particles from which we have derived our physical models. The moving samples are actually blobs of matter and the equations governing their evolution are different from the mechanical equations governing elementary particle systems. In the remainder of this thesis therefore we will refer to these samples as *blobs*.

4.1 Smoothing Kernels

The color sensations which give the effect of light are abstractions which do not permit me to cover my canvas or to continue the delimitation of the objects when the points of contact are tenuous and delicate. *Paul Cézanne*

In most numerical methods the solution of an equation is obtained by sampling the physical quantities at a given scale. In many cases, such as in the finite difference method, the quantities are sampled on a grid. In this case the method can only resolve scales which are bigger than the grid spacing. Let $\mathbf{f}(\mathbf{x},t)$ be a function modelling the physical properties of a given phenomenon. Most numerical methods compute a solution for the smoothed version $\mathbf{f}_\sigma(\mathbf{x},t)$ of the function. Such a smoothing can be represented in general as a convolution with a shift invariant *smoothing kernel* $W(\mathbf{r},\sigma)$, where σ is the smallest scale resolved by the numerical method:

$$\mathbf{f}_\sigma(\mathbf{x},t) = \int_{\mathbf{R}^3} W(\mathbf{x}-\mathbf{x}',\sigma)\mathbf{f}(\mathbf{x}',t)\,d\mathbf{x}'. \tag{4.1}$$

The smoothing kernel $W(\mathbf{r},\sigma)$ is normalized usually so that its integral over the whole space is unity, and tends towards a delta distribution for the scale σ tending to zero. The latter condition is required for the original function $\mathbf{f}$ to be recovered:

$$\mathbf{f}_\sigma(\mathbf{x},t) \to \mathbf{f}(\mathbf{x},t), \qquad \text{as} \quad \sigma \to 0.$$

An important example of a kernel having these properties is the *Gaussian smoothing kernel* defined by:

$$G_\sigma(\mathbf{r}) = \frac{1}{(2\pi)^{3/2}\sigma^3}\exp\left(-\frac{|\mathbf{r}|^2}{2\sigma^2}\right). \tag{4.2}$$

If the smoothing kernel only depends on the magnitude of its argument and is sufficiently differentiable, as is the case for the Gaussian kernel, then the smooth approximation is accurate up to $O(\sigma^2)$. In general, if a spherically symmetric kernel has zero moments up to an order p, i.e., when

$$\int_0^\infty r^q W(r,\sigma)\,dr = 0, \qquad 0 < q \leq p,$$

then the approximation is of order $p+1$. This follows directly from a Taylor expansion of the function $\mathbf{f}$ in its spatial variable:

$$\begin{aligned}
\mathbf{f}_\sigma(\mathbf{x},t) &= \int_0^\infty \int_{|\mathbf{s}|=r} W(r,\sigma)\left(\mathbf{f}(\mathbf{x},t) + r\mathbf{f}'(\mathbf{x},t) + \frac{r^2}{2}\mathbf{f}''(\mathbf{x},t) + \cdots\right) d\mathbf{s}\,dr \\
&= \mathbf{f}(\mathbf{x},t) + 4\pi\frac{\mathbf{f}^{(p+1)}(\mathbf{x},t)}{(p+1)!}\int_0^\infty W(r,\sigma)r^{p+3}\,dr = \mathbf{f}(\mathbf{x},t) + O(\sigma^{p+1}).
\end{aligned}$$

The last equality follows directly when the smoothing kernel is a properly scaled version of a smoothing kernel W_1 of width 1:

$$W(r,\sigma) = \frac{1}{\sigma^3}W_1\left(\frac{r}{\sigma}\right).$$

The Gaussian smoothing kernel, for example, is obtained in this fashion. The integral involved in the $(p+1)$-th term is then calculated using the change in variable $u = r/\sigma$:

$$\int_0^\infty W(r,\sigma)r^{p+1}\,dr = \int_0^\infty \frac{1}{\sigma^3}W_1(u)\sigma^{p+3}u^{p+3}\,\sigma du = \sigma^{p+1}\int_0^\infty W_1(u)u^{p+3}\,du.$$

Since the last integral is finite, the term is of order $p+1$ in σ.

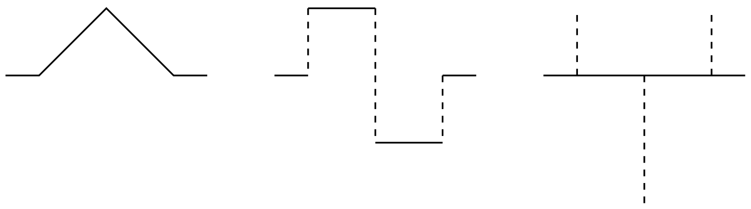
Figure 4.1: The triangle function (left), its first derivative (middle) and its second derivative (right).

Consequently, for even smoothing kernels (of order 1), $\mathbf{f}(\mathbf{x},t)$ can always be replaced by its smoothed equivalent to within the order of accuracy of the smoothing process itself. For example, this implies that

$$(\mathbf{f}(\mathbf{x},t)\mathbf{g}(\mathbf{x},t))_\sigma = \mathbf{f}_\sigma(\mathbf{x},t)\mathbf{g}_\sigma(\mathbf{x},t) + O(\sigma^2). \tag{4.3}$$

The same is true for operations such as differentiation:

$$\left(\nabla\mathbf{f}(\mathbf{x},t)\right)_\sigma = \nabla\mathbf{f}_\sigma(\mathbf{x},t) + O(\sigma^2). \tag{4.4}$$

Consider the finite difference method on the real line. The values of the function f are reconstructed usually by linear interpolation of neighbouring points of the subdivision. This interpolation corresponds to a smoothing with a triangular kernel:

$$W_T(r,\sigma) = \begin{cases} 1 - |r|/\sigma, & |r| \le \sigma \\ 0 & \text{otherwise} \end{cases}$$

Let x_i denote the points of the finite difference subdivision on which the field is sampled. The integral of Equation 4.1 then reduces to a sum:

$$f_\sigma(x,t) = 2\sigma\sum_{i=1}^{N} f(x_i,t)W_T(x - x_i,\sigma). \tag{4.5}$$

The usual finite difference approximations for the gradient and the Laplacian naturally follow from this formulation. For example, the second derivative of the smoothed field at a sample x_j becomes a central difference:

$$\frac{\partial^2}{\partial x^2}f_\sigma(x_j,t) = \sum_{i=1}^{N} f(x_i,t)\frac{\partial^2}{\partial x^2}W_T(x_j - x_i,\sigma) = \frac{-2f(x_j) + f(x_{j-1}) + f(x_{j+1})}{\sigma^2}.$$

This follows directly from the fact that the "second-derivative" of a triangle function is equal to three delta impulses at the discontinuities (see Figure 4.1). Thus we see that we can define differential operators on a set of samples by applying them to the smoothing kernel. Interestingly, this procedure is used in the *theory of distributions* [108] which generalizes the concepts and results of calculus to distributions. Distributions are defined on a space of *test functions* rather than on the reals. The delta distribution, for example, maps each function to its value at the origin. The derivative of a distribution is then defined as the distribution evaluated at the derivative of the test function. Consequently the derivative of the delta distribution is equal to the derivative of the test function evaluated at the origin:

$$\frac{d\delta}{dx}\{f\}(x) = \delta\left\{\frac{df}{dx}\right\} = \frac{df}{dx}(0).$$

These ideas are generalized to physical quantities sampled using more general smoothing kernels in the next section.

4.2 The Blob Representation

4.2.1 Basic Model

Let $\mathbf{x}_1(t), \cdots, \mathbf{x}_N(t)$ be N arbitrarily located samples varying over time. To each sample we can associate a mass of matter $m_i(t)$ and consequently, we define a *sample mass density* by

$$\rho(\mathbf{x}, t) = \sum_{i=1}^{N} m_i(t)\delta(\mathbf{x} - \mathbf{x}_i(t)). \tag{4.6}$$

The smoothed version at a scale σ of this density is then given by:

$$\rho_\sigma(\mathbf{x}, t) = \sum_{i=1}^{N} m_i(t) W(\mathbf{x} - \mathbf{x}_i(t), \sigma). \tag{4.7}$$

In other words, the smoothing has the effect of smearing out the mass of each sample in space. To each sample we can thus associate a blob density of mass:

$$B_i(\mathbf{x}, t, \sigma) = m_i(t) W(\mathbf{x} - \mathbf{x}_i(t), \sigma).$$

In particular, the scaled blob B_i/M, where $M(t)$ is the total mass, can be interpreted as the probability distribution of finding a particle belonging to the blob at location $\mathbf{x}$ and time t. The smoothing length in this case gives the uncertainty we have about the blob. In general we can assume that this uncertainty is specific to each blob and hence $\sigma \rightarrow \sigma_i$. Representations similar to that of the density given in Equation 4.7 can be obtained for any other physical property described by a function $\mathbf{f}$ using Equation 4.6 again:

$$(\rho\mathbf{f})_\sigma(\mathbf{x}, t) = \int \mathbf{f}(\mathbf{x}', t)\rho(\mathbf{x}', t) W(\mathbf{x} - \mathbf{x}', \sigma)\, d\mathbf{x}' = \sum_{i=1}^{N} m_i(t)\mathbf{f}_i(t) W(\mathbf{x} - \mathbf{x}_i(t), \sigma),$$

where $\mathbf{f}_i(t) = \mathbf{f}(\mathbf{x}_i(t), t)$. The sum on the right hand side can be interpreted as a Monte-Carlo estimate of the integral and the error is therefore $O(1/\sqrt{N})$. In practice, when the particles are nearly equidistributed, the error is more like $O((\log N)^3/N)$ [59]. Using Equation 4.3, we obtain an expression for the smoothed function $\mathbf{f}$:

$$\mathbf{f}_\sigma(\mathbf{x}, t) = \frac{1}{\rho_\sigma(\mathbf{x}, t)} \sum_{i=1}^{N} m_i(t)\mathbf{f}_i(t) W(\mathbf{x} - \mathbf{x}_i(t), \sigma). \tag{4.8}$$

Derivation of another expression can be obtained if we consider that ρ is approximately equal to its smoothed version and therefore use Equation 4.3:

$$\mathbf{f}_\sigma(\mathbf{x}, t) = (\mathbf{f}\rho/\rho_\sigma)_\sigma(\mathbf{x}, t) = \sum_{i=1}^{N} \frac{m_i(t)}{\rho_i(t)} \mathbf{f}_i(t) W(\mathbf{x} - \mathbf{x}_i(t), \sigma), \tag{4.9}$$

where $\rho_i(t) = \rho(\mathbf{x}_i(t), t)$.

4.2.2 Multi-Scale Blob Models

If we start from the bottom of a tree-covered slope and gradually move away from it, it will undergo a continuous change in aspect. Beyond a certain distance we discover that the change is not merely in size but in sharpness and hue. It is difficult, however, to determine just where the visual transformation took place. We can retrace our steps and repeat the process, but this is likely only to add to our confusion. *T. Higuchi*

In this section we describe how a hierarchical model of blobs at different scales can be constructed from the basic blob representation. In many applications we need a description of a phenomenon at various scales of detail. From the initial representation, we build a binary tree data structure. At each level of detail, half as many blobs are considered, and the tree is therefore of depth $\lceil \log N \rceil$. The algorithm is straightforward. At each larger scale, adjacent blobs are grouped together into a new blob. The new smoothing scale at each level is application dependent but is usually bigger than the scale of the finer level. The following algorithm uses two routines called "GetNearest(b,L)" and "GroupBlobs(b,b')". The former returns the blob nearest to the blob b in the set L. The function GroupBlobs groups two blobs b and b' together into a "bigger" blob that is returned by the function. The following algorithm then computes the tree-like hierarchical data structure:

```
For $i \in \{1, \cdots, N\}$ do
    $b_i \rightarrow$left$=$ $b_i \rightarrow$right$= 0$
end for
$L = \{b_1, \cdots, b_N\}$
Do
    $L' = \emptyset$
    while $L \neq \emptyset$ do
        $b =$ first element of the set $L$
        $L = L - \{b\}$
        $b' =$ GetNearest$(b, L)$
        $L = L - \{b'\}$
        $b'' =$ GroupBlobs$(b, b')$
        $b'' \rightarrow$left$= b$
        $b'' \rightarrow$right$= b'$
        $L' = L' \cup \{b''\}$
    end while
    $L = L'$
until $|L| = 1$
```

Let $f(K)$ with $K = |L|$ denote the complexity of the function "GetNearest". In order to simplify the following analysis, we assume that $N = 2^d$. The complexity of the algorithm is then given by

$$\begin{aligned} T(N) &= f(N) + f(N-2) + f(N-4) + \cdots + f(2) + \\ & \quad f(N/2) + f(N/2-2) + \cdots + f(2) + \\ & \quad \vdots \\ & \quad f(4) + f(2) + \end{aligned}$$

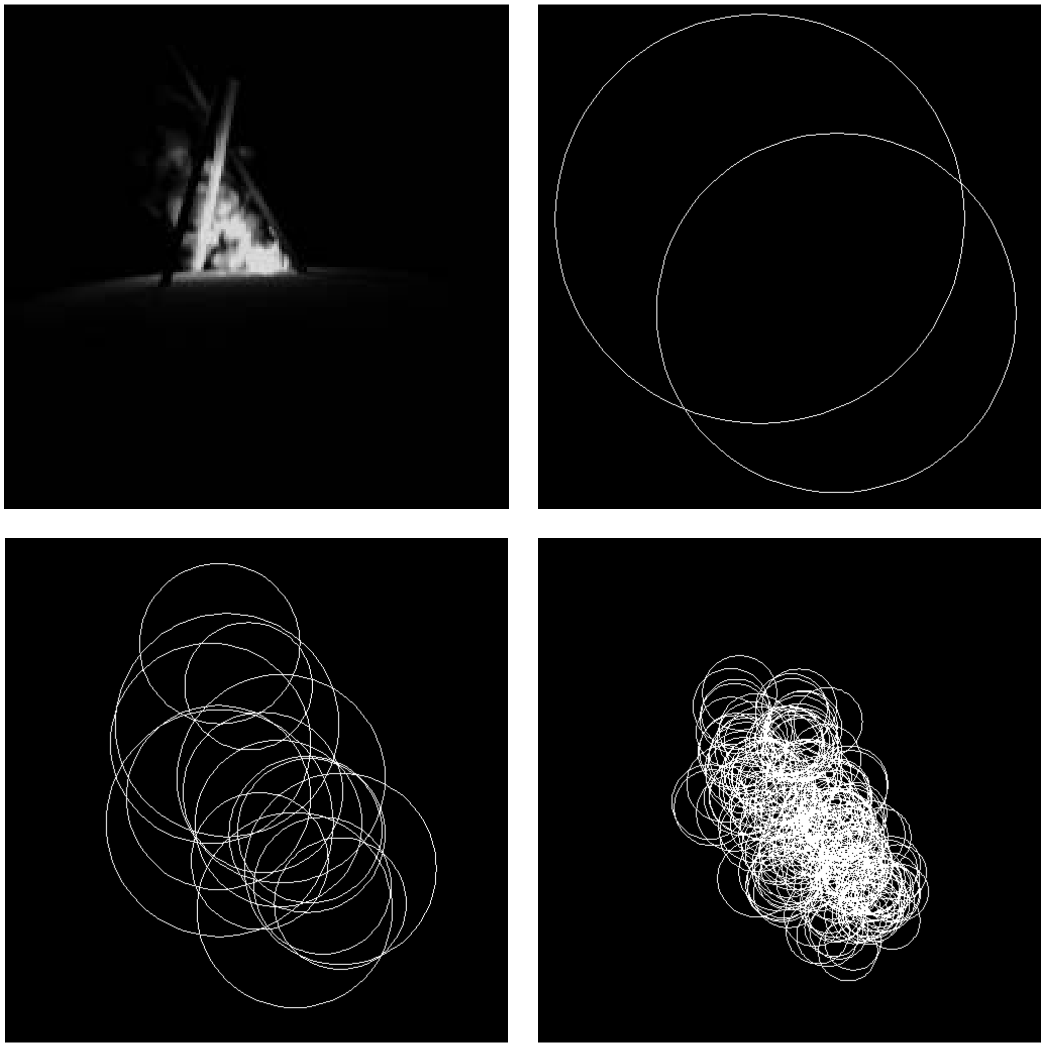

Figure 4.2: Multi-scale blob representation of a Bonfire. From top to bottom and left to right: (a) rendered version of the fire, (b) 2 blobs, (c) 16 blobs and (d) 256 blobs.

$$f(2)$$
$$= \sum_{k=1}^{d} \sum_{l=1}^{2^{k-1}} f(2l)$$

When straightforward exhaustive nearest neighbour search is used, the complexity of "Get-Nearest" is $f(K) = K$ and the complexity of the algorithm is:

$$T(N) = \sum_{k=1}^{d} 2^{k-1}(2^{k-1}+1) = \sum_{k=1}^{d} 4^{k-1} + \sum_{k=1}^{d} 2^{k-1} = \frac{N^2}{3} + N - \frac{4}{3} = O(N^2).$$

In general, this is prohibitively expensive. A more efficient algorithm is obtained by storing the blobs in a grid at each level and by considering blobs in neighbouring cells only. When the blobs are distributed uniformly, we can assume that the number of blobs in each grid cell is independent of the number of blobs. In this situation, the complexity of the nearest neighbour search is equal to a constant f_0. Consequently, the complexity of the algorithm is:

$$T(N) = \sum_{k=1}^{d} (2^{k-1} - 1) f_0 = (2^d - 1 - d) f_0 = (N - 1 - \log N) f_0 = O(N).$$

We have implemented both algorithms and have found that in most practical situations, the second method is an order of magnitude faster than the first method. This is good evidence for our analysis. One problem with the second approach, however, is that it requires a three-dimensional grid. In practice, we first attempt to allocate enough memory for a grid of spacing σ. If this fails, we must then build the tree using the more expensive approach. Figure 4.2 depicts different levels of the multi-scale representation of a bonfire modelled using 225 blobs.

4.3 Numerical Blob Methods

We now apply the blob representation to the approximate solution of diffusion and advection-diffusion equations. The algorithms we present have not appeared before and are another contribution of this thesis. Although the accuracy of the algorithms may not be high enough for applications in other disciplines, their effectiveness in many areas of computer graphics is demonstrated in this thesis. In the last subsection of this chapter we mention similar work done in computational fluid dynamics. These algorithms can therefore be applied to the animation of water.

4.3.1 Steady State Diffusion Equations

In many situations we are interested in the steady state of the diffusion equation, i.e., $\frac{\partial U}{\partial t} = 0$ in Equation 2.5 and therefore $U(\mathbf{x}, t) \rightarrow U(\mathbf{x})$. This is the case, for example in the diffusion approximation of the propagation of light through a participating medium. The steady-state of Equation 2.5 (with $\mathbf{F}^{ext} = \mathbf{0}$) corresponds to

$$\nabla \cdot (\kappa(\mathbf{x}) \nabla U(\mathbf{x})) - \alpha(\mathbf{x}) U(\mathbf{x}) + S(\mathbf{x}) = 0. \qquad (4.10)$$

We can solve this equation by interpreting the set of blobs $\{B_i(\mathbf{x}, \sigma_i)\}$ as a finite element basis (see Section 2.3.1). Note that we have also allowed for different smoothing scales. First we observe that the diffusion operator is actually given by

$$\nabla \cdot (\kappa(\mathbf{x}) \nabla U(\mathbf{x})) = \nabla \kappa(\mathbf{x}) \cdot \nabla U(\mathbf{x}) + \kappa(\mathbf{x}) \nabla^2 U(\mathbf{x}).$$

Then, by using a blob representation for both the function $U(\mathbf{x})$ and the source term $S(\mathbf{x})$:

$$U_\sigma(\mathbf{x}) = \frac{1}{\rho_\sigma(\mathbf{x})} \sum_{i=1}^{N} U_i B_i(\mathbf{x}, \sigma_i)$$
$$S_\sigma(\mathbf{x}) = \frac{1}{\rho_\sigma(\mathbf{x})} \sum_{i=1}^{N} S_i B_i(\mathbf{x}, \sigma_i),$$

and substituting them back into Equation 4.10 we get that

$$\sum_{i=1}^{N} U_i \left(\nabla \kappa(\mathbf{x}) \cdot \nabla B_i(\mathbf{x}, \sigma_i) + \kappa(\mathbf{x}) \nabla^2 B_i(\mathbf{x}, \sigma_i) - \alpha(\mathbf{x}) B_i(\mathbf{x}, \sigma_i) \right) + S_i B_i(\mathbf{x}, \sigma_i) = 0. \qquad (4.11)$$

where we have used Equations 4.3 and 4.4 when applicable. A system of N linear equations is obtained by either evaluating Equation 4.11 at the centre of the blobs $\mathbf{x}_j$ (collocation) or by multiplying the equation by a blob B_j and integrating it over the entire space (Galerkin method). The former gives us a set of N linear equations which can be solved directly, if N is not too large, using an LU-decomposition like [77]. This method, however, becomes unstable if the centres $\mathbf{x}_j$ are very proximate.

In the Galerkin method, the integrals of the product of two basis functions have to be calculated. When the smoothing kernel is Gaussian (Equation 4.2), these integrals can be computed exactly:

$$\begin{aligned} G_{\sigma_i,\sigma_j} &= \int_{\mathbf{R}^3} G_{\sigma_i}(\mathbf{x}-\mathbf{x}_i) G_{\sigma_j}(\mathbf{x}-\mathbf{x}_j)\, d\mathbf{x} \\ &= G_{\sigma_i}(\mathbf{x}) * G_{\sigma_j}(\mathbf{x}-(\mathbf{x}_j-\mathbf{x}_i)) \\ &= G_{\sqrt{\sigma_i^2+\sigma_j{}^2}}(\mathbf{x}_j-\mathbf{x}_i). \end{aligned}$$

The Galerkin method also requires the evaluation of the integral of a basis function multiplied by the gradient and the Laplacian of another basis function, respectively. Again for the case of a Gaussian smoothing kernel, these integrals can be computed:

$$\begin{aligned} \nabla G_{\sigma,\sigma_i} &= \int_{\mathbf{R}^3} \nabla G_{\sigma_i}(\mathbf{x}-\mathbf{x}_i) G_{\sigma_j}(\mathbf{x}-\mathbf{x}_j)\, d\mathbf{x} = \nabla G_{\sqrt{\sigma_i^2+\sigma_j^2}}(\mathbf{x}_j-\mathbf{x}_i), \\ \nabla^2 G_{\sigma_i,\sigma_j} &= \int_{\mathbf{R}^3} \nabla^2 G_{\sigma_i}(\mathbf{x}-\mathbf{x}_i) G_{\sigma_j}(\mathbf{x}-\mathbf{x}_j)\, d\mathbf{x} = \nabla^2 G_{\sqrt{\sigma_i^2+\sigma_j^2}}(\mathbf{x}_j-\mathbf{x}_i). \end{aligned}$$

Hence, Equation 4.11 is given by the following N linear equations:

$$\begin{aligned} \sum_{i=1}^{N} U_i m_i \Big(\nabla\kappa(\mathbf{x}) \cdot \nabla G_{i,j} + \kappa(\mathbf{x})\nabla^2 G_{i,j} - \alpha(\mathbf{x}) G_{i,j}\Big) & \\ + S_i m_i G_{i,j} = 0, \qquad j = 1,\cdots,N, & \end{aligned}$$

where $G(i,j) = G_{\sqrt{\sigma_i^2+\sigma_j^2}}(\mathbf{x}_j-\mathbf{x}_i)$. In this case the Galerkin method is actually equivalent to a collocation method with bigger blobs. This method is therefore also prone to numerical instabilities when it is solved by matrix inversion.

4.3.2 Advection-Diffusion Equations

We now turn to the problem of solving the more general advection-diffusion equations. Let us rewrite Equation 2.5 by assuming that the diffusion rate κ is constant and, for notational convenience, that $v = 1$ and $m = 1$:

$$\frac{\partial U}{\partial t} + \mathbf{F}^{ext} \cdot \nabla U = \kappa \nabla^2 U - \alpha U + S. \tag{4.12}$$

Let $\mathbf{x}(t)$ denote the motion of a particle satisfying:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}^{ext}(\mathbf{x}(t), t), \qquad \mathbf{x}(0) = \mathbf{x}_0.$$

In other words the trajectory is a streamline of the force field $\mathbf{F}^{ext}$ (this trajectory should not be confused with dynamics of a point particle subjected to the force field). The variation of the physical quantity U on this particle becomes a function of time only and its time derivative is equal to

$$\frac{d}{dt}U(\mathbf{x}(t),t) = \frac{\partial}{\partial t}U(\mathbf{x}(t),t) + \frac{d}{dt}\mathbf{x}(t)\cdot\nabla U(\mathbf{x}(t),t) = \frac{\partial}{\partial t}U + \mathbf{F}^{ext}\cdot\nabla U.$$

Equation 4.12 can thus be rewritten for the evolution of the function $U(t)$ on a particle as

$$\frac{d}{dt}U = \kappa\nabla^2 U - \alpha U + S.$$

This derivative is often called the *total derivative* of a function and is sometimes denoted by $\frac{D}{Dt}$. The source term is modelled by generating new blobs over time at locations according to the probability density φ_S obtained from the source term:

$$\varphi_S(\mathbf{x},t) = S(\mathbf{x},t) \Big/ \left(\int_{\mathbf{R}^3} S(\mathbf{x}',t)\, d\mathbf{x}'\right).$$

Generally, the source term is constant on a given region and zero elsewhere so that we can generate the locations directly from a uniform distribution. The rate of creation of new blobs can be controlled either by fixing the initial value $U_i(0)$ of the function or by fixing the number N_0 of new blobs at each time step. For a given time step Δt the two are related by:

$$N_0 U_i(0) = \Delta t\ S(\mathbf{x}_i,t).$$

In addition with each new blob we assign a fixed smoothing scale $\sigma(0)$ provided by the user. Once the blobs are created their evolution is governed by the advection diffusion equation without the source term. In particular, by using a blob representation for the function $U(\mathbf{x},t)$ we can require that this equation has to be satisfied by each blob:

$$\frac{d}{dt}B_i = \kappa\nabla^2 B_i - \alpha B_i, \qquad i = 1,\cdots,N.$$

The diffusion term can be satisfied by a Gaussian blob moving along the force field by increasing the smoothing scale σ over time. Indeed, diffusion over a time t at a constant rate κ is equivalent to a convolution with a Gaussian having a standard deviation equal to $\sqrt{\kappa t}$ [105]:

$$G_\sigma(\mathbf{y}-\mathbf{x}_i(t)) * G_{\sqrt{\kappa t}}(\mathbf{y}-\mathbf{x}) = G_{\sqrt{\sigma^2+\kappa t}}(\mathbf{x}-\mathbf{x}_i(t)).$$

The diffusion term is therefore satisfied, if the smoothing scale increases over time as

$$\sigma(t) = \sqrt{\sigma(0)^2 + \kappa t}.$$

The loss equation for U at the center of each blob becomes:

$$\frac{d}{dt}\left(\frac{U_i(t)m_i(t)}{\rho(\mathbf{x}_i(t),t)}\right) = \frac{d}{dt}U_i(t)V_i(t) = -\alpha U_i(t)V_i(t),$$

where $V_i(t) = m_i(t)/\rho_i(\mathbf{x}_i(t),t)$ is the volume of the blob. For Gaussian blobs this is given by $V_i(t) = (2\pi)^{3/2}\sigma(t)^3$. The solution of the above equation is an exponential decay with time:

$$U_i(t) = U_i(0)\left(\frac{\sigma(0)}{\sigma(t)}\right)^3 e^{-\alpha t}.$$

The advection-diffusion equation is satisfied then by the function U whose coefficients are updated over time using the following algorithm:

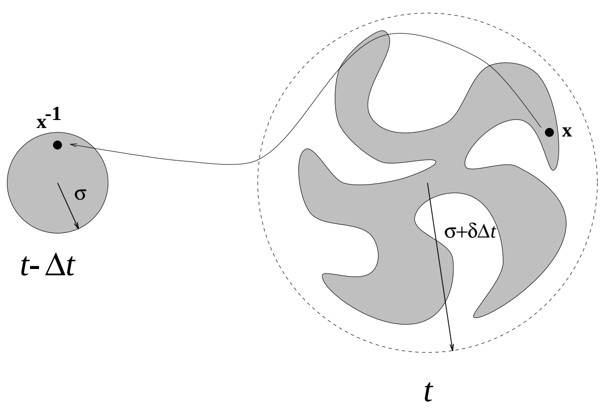

Figure 4.3: Inverse warping of a blob: the density at each point is obtained by backtracing it over time through the wind field and evaluating the smoothing kernel.

```
$t = 0$
Fix number of new blobs created at each time step $N_0$
and their initial spread $\sigma(0)$.
Do
    for $i = 1, \ldots, N_0$ do                    { generate new blobs }
        Generate a point $\mathbf{x}'$ according to the probability density $\varphi_S$
        Set initial value of the function: $U_i = \Delta t S(\mathbf{x}_i, t)/N_0$
        also initialize $V_i = \sigma^3$
    end for
    $N = N + N_0$
    for $i = 1, \ldots, N$ do
        $\mathbf{x}_i = \mathbf{x}_i + \Delta t\ \mathbf{F}^{ext}(\mathbf{x}_i, t)$        { advect centre of blob }
        $\sigma_i^2 = \sigma_i^2 + \kappa \Delta t$                    { increase size of blob }
        $U_i = U_i V_i - \alpha \Delta t\ U_i V_i$               { update value of the function }
        $V_i = \sigma^3$
        $U_i = U_i / V_i$
        if $U_i <$ EPS then kill blob
    end for
    $t = t + \Delta t$
Until *Bored*
```

The method is therefore of $O(N)$ for each time step. The accuracy and speed of the algorithm is controlled by the time step Δt, the number of new blobs at each time step N_0 and the initial smoothing scale $\sigma(0)$.

One problem with this method of solution is that as the blobs get bigger through the action of diffusion, significant resolution is lost. Indeed, the advection of a blob is actually a complicated warped blob as shown in Figure 4.3. A possible solution to this problem would be to advect a fixed number of samples associated with each blob and then to reconstruct the blob from these samples. For example, when the blob size is bigger than a certain threshold, we can split the blob into a fixed number of new blobs. Another approach is to backtrace through the advecting field from the warped blob toward the initial Gaussian blob (see Figure 4.3). Indeed, for each point $\mathbf{x}$ on the warped blob there corresponds a point $\mathbf{x}^{-1}$ on the initial blob. This point is calculated by integrating along the force field backward in time:

$$\mathbf{x}^{-1} = \mathbf{x} - \int_t^{t_i} \mathbf{F}^{ext}(\mathbf{x}(s), s)\, ds \quad \text{and} \quad \mathbf{x}(t) = \mathbf{x}. \tag{4.13}$$

where t_i is the creation time of the blob. The density of the blob at a point $\mathbf{x}$ is then defined as the initial blob evaluated at the point $\mathbf{x}^{-1}$:

$$G_\sigma(\mathbf{x} - \mathbf{x}_i) \rightarrow G_\sigma\left(\mathbf{x}^{-1} - \mathbf{x}_i^{-1}\right).$$

The cost of evaluating the integral in Equation 4.13 grows as the time interval $[t_k, t]$ increases. Therefore, this method is effective for modelling blobs which have a short life-time only, i.e., when the absorption rate α is high. This problem is addressed in [56] by integrating only for a fixed time, and then continuously fade out the blob and fade in an undistorted blob which immediately begins distorting again. The warping method can also be used to slightly modify the spherical shape of the Gaussian. In this case, we only backtrace each point for a small time interval δt:

$$\mathbf{x}^{-1} = \mathbf{x} - \int_t^{t-\delta t} \mathbf{F}^{ext}(\mathbf{x}(s), s)\, ds.$$

This corresponds to an instantaneous warping by the field surrounding the blob. In particular, when the integral in the above equation is computed using only one sample, the transformation is given by evaluating the advecting field:

$$\mathbf{x}^{-1} = \mathbf{x} - \delta t \mathbf{F}^{ext}(\mathbf{x}, t).$$

In practice, the back warping operation is done by subdividing the warping interval into K subintervals of size Δs. For a given interval of time $[a, b]$, the warping is performed by the following algorithm.

```
WarpBack([$a, b$],$\mathbf{x}$):
    $K = (b - a)/\Delta s$
    $\mathbf{y} = \mathbf{x}$
    $s = b$
    for $k = 1, \cdots, K$ do
        $\mathbf{y} = \mathbf{y} - \Delta s \mathbf{F}^{ext}(\mathbf{y}, s)$
        $s = s + \Delta s$
    end for
    return $\mathbf{y}$
```

The complexity of the algorithm $O(K)$ is directly related to the precision Δs desired and the length of the interval $b - a$.

In order to evaluate the function $U(\mathbf{x}, t)$ more efficiently, the domains of the Gaussians are truncated beyond a certain radius R. The radius can be related to a certain precision ϵ by inverting the Gaussian (see Equation 4.2):

$$R^2 = -6\sigma^2 \log\left(\sqrt{2\pi}\sigma\epsilon\right).$$

When the blobs are distributed fairly uniformly, the sum of the blobs at a certain location can be computed more efficiently by storing the blobs in a grid. By setting the size of each grid cell to that of a sphere of radius R, the sum can be evaluated up to a precision

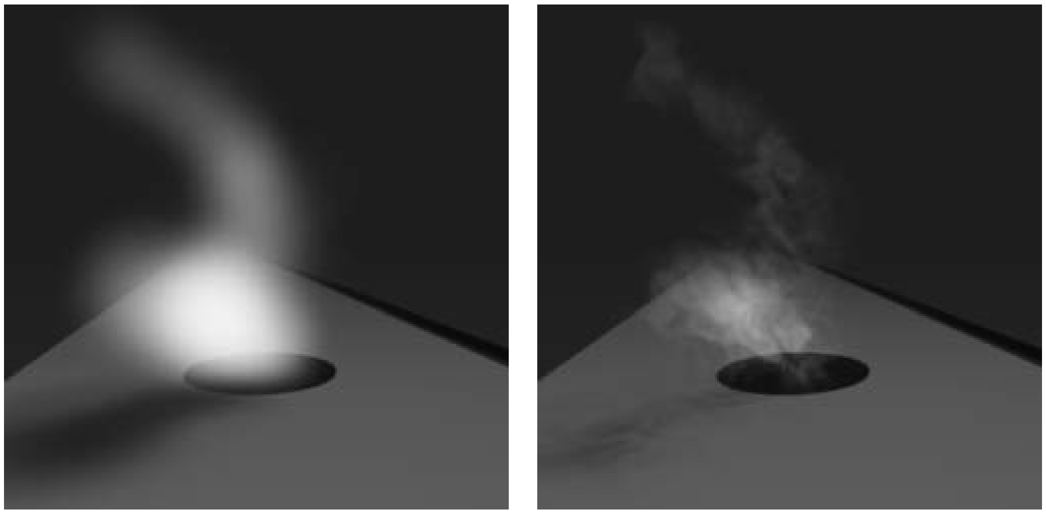

Figure 4.4: Spherical versus warped blobs. The steam on the left is rendered without warping while the steam on the right is obtained by backtracing the points for 10 time steps through the wind field.

ϵ by considering blobs lying in neighbouring cells only. Therefore, the truncation induces a bounding sphere for each Gaussian. A bounding sphere for the initial Gaussian blob defines a larger bounding sphere for the warped blob as illustrated in Figure 4.3. The size of this new bounding sphere can be estimated when the advecting field is a white noise (see Section 3.3.1). In this case the paths of the particles correspond to realizations of Brownian motion. The standard deviation of the increments of Brownian motion is equal to [106]:

$$\sigma'(t) = \langle |\mathbf{x}(t_i) - \mathbf{x}(t)| \rangle = \sqrt{\langle |\mathbf{F}^{ext}| \rangle (t - t_i)}.$$

Hence by using *Tchebyshev's inequality*

$$P(|\mathbf{x} - \bar{\mathbf{x}}| \geq a) \leq \frac{\langle |\mathbf{x} - \bar{\mathbf{x}}|^2 \rangle}{a^2},$$

the warped blob lies with probability $1 - a$ in the sphere centred at $\mathbf{x}_i$ and of radius:

$$R = \sigma'(t) a^{-1/2}.$$

For example, a 99% probable bounding sphere is given by an increase in the radius corresponding to $10\sqrt{t - t_i}$. For certain deterministic fields, the bounding radius can be computed exactly. An algorithm to evaluate the function U is given next.

```
$U = 0$
$\rho = 0$
for $i = 1, \cdots, N$ do
    if $|\mathbf{x} - \mathbf{x}_i| > \sigma(t) + \sigma'(t)$ next $i$.
    $\mathbf{x}^{-1}$ = WarpBack($[t_i, t]$,$\mathbf{x}$)
    $B = m_i(t) G_{\sigma(t)} \left( \mathbf{x}^{-1} - \mathbf{x}_i^{-1} \right)$
    $\rho = \rho + B$
```

```
        $U = U + U_i(t)B$
    end for
    $U = U/\rho$
```

The algorithm can be improved if intermediate results are stored in the warping procedure. In particular, if all blobs were created at the same time, the warping calculation has to be done only once and can therefore be taken out of the sum. The effects of the warping are illustrated in Figure 4.4. The picture on the left depicts the use of unwarped blobs which are allowed to expand uniformly in all directions. At right, we see a much more convincing depiction in which the blobs are themselves warped due to advection. Only $K = 10$ integration steps were used to backwarp in time.

4.3.3 Hydrodynamic Equations

[Water]. Always in motion, ever-flowing (whether at steam rate or glacier speed), rhythmic, dynamic, ubiquitous, changing and working its changes, a mathematics turned wrong side out, a philosophy in reverse, the ongoing odyssey of water is virtually irresistable. *Tom Robbins*

Following, we briefly mention similar work done in the area of fluid dynamics. In *Smoothed Particle Hydrodynamics* (SPH) the blob representation given in Equation 4.9 is used to represent the the velocity field and the temperature field. The first SPH methods were developed for the Navier-Stokes equations with zero viscosity. The continuity equation for the density of the fluid:

$$\frac{d}{dt}\rho = \frac{\partial}{\partial t}\rho + \mathbf{u}\cdot\nabla\rho = 0.$$

is naturally satisfied if the blobs are advected along the velocity field $\mathbf{u}(\mathbf{x}, t)$ of the fluid. The velocity of the fluid is resolved in terms of the the pressure $p(\mathbf{x}, t)$ using the following relation:

$$\frac{1}{\rho}\nabla p = \nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho.$$

Using Equation 4.3, the change in velocity on a blob advected by it is:

$$\frac{d}{dt}\mathbf{u}_i = -\sum_{j=1}^{N} m_j\left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right)\nabla W(\mathbf{x}_i - \mathbf{x}_j, \sigma). \tag{4.14}$$

A similar expression is obtained for the temperature of the fluid [60]. The pressure is calculated through a state equation involving both the density and the temperature. Viscosity is usually modelled by inserting an ad hoc term on the right hand side of Equation 4.14. This method has been applied mainly to the simulation of compressible fluids in astrophysical problems. In fact, it was first reported in an astrophysical journal [22]. However, this procedure can be applied to nearly incompressible fluids such as water, by slowing down the simulation of a compressible fluid. It does, however, become more expensive as smaller time steps are needed. Monaghan [61] applied this method to the simulation of waves and bursting dams in two dimensions, and there is the potential for this technique to be applied to the simulation of water flows in computer graphics.

For incompressible fluids, the hydrodynamic equation can be written in terms of the vorticity $\Omega = \nabla\times\mathbf{u}$ only. A procedure known as the *blob vortex method* approximates the vorticity field by a set of blobs [46]. This method fundamentally is restricted to two-dimensional flows, where each vortex is defined entirely by its location in the plane only, and can therefore be approximated by a set of point blobs. In three dimensions, the vorticity is defined by both its position and its direction, and the representation becomes considerably more complicated.

Chapter 5

Rendering of Density Fields

Psychologically it does not make much sense to say that we "copy" what we see in the visible world. What we see extends in depth, while our painting surface is flat. The elements of what we see differ in colour. To invent a code of colour combinations distributed on a plane for the variety of experience in the real world is, of course, the achievement of naturalism. It is an achievement simply because the real world does not look like a flat picture, though a flat picture can be made to look like the real world. *E. H. Gombrich*

Gaseous phenomena such as steam, clouds and fire, are visible to us by modulating the intensity of light. For example, through energy emission the shape and motion of a fire are discernible. Clouds and steam, on the other hand, are observable by scattering light from the sun or from a spotlight into the direction of our eyes. These scattering effects are very complicated in general. Indeed, light coming from the sun may reach our eyes after an arbitrary number of scattering events through a cloud. The computation of the interactions of light with an arbitrary participating medium is, therefore, a complex task. For this reason, we will use a general formulation based on a transport theoretic model for the propagation of light. In order to get tractable solutions, we will introduce suitable approximations into the theory. We will state these approximations, indicating the implied limitations. We do not seek simulations which are physically accurate, but rather focus on visually convincing and consistent depictions that are motivated by physical models. The results, therefore, will not be compared with radiances obtained from actual measurements. The chapter is organized as follows. In the first section, we present the transport theory of radiative transfer on which all our algorithms are based. In the second section, we state the approximate representation of the intensity used in our algorithms. In the third section, we introduce a general algorithm to account for the interchange of intensities in an environment composed of diffuse surfaces and single scatterers. We discuss an approximate solution for the effects of multiple scattering using a diffusion model in the Section 5.4. Then, in section 5.6 we present a new rendering paradigm called *stochastic rendering* and apply it to the efficient rendering of density fields. Finally, in the last section we present an algorithm to render hair modelled as a set of fuzzy segments.

5.1 Radiative Transfer

5.1.1 The Intensity Field and Domain of Applicability

In chapter 2, we introduced a kinetic description of the intensity of light modelled as a set of photon particles. The intensity of light is then defined from the one-particle density as (see Section 2.2.1):

$$I_\nu(\mathbf{x}, \mathbf{s}, t) = (h\nu)c\varphi_1(\mathbf{x}, \mathbf{s}, \nu, t).$$

Here, the energy of the photon is replaced by its frequency, through the relation $E = h\nu$. Furthermore, by neglecting refraction effects, the velocity of each photon is equal to the speed of light c. The units of the intensity field so defined are watts per unit area per unit solid angle, in accordance with the radiometric definition of *radiance*. The intensity field is therefore interpreted as an "energy angular flux". We assume that there is no interaction between the frequencies of the intensity field. This implies that the intensity field can be resolved for each frequency independently. The relevant equation describing the evolution of the intensity field is then given by the one-speed linear transport equation (Equation 2.3). In general, this description is reasonable when quantum and diffraction effects, polarization and interference are negligeable. In addition, it is assumed that the optical properties of the participating medium are not modified by the intensity field. Even under these restrictive conditions, the transport theoretic description of light is rich enough to model a broad range of visual effects. Indeed, it forms the basis of most rendering algorithms in computer graphics [33]. In the remainder of the presentation, we use the wavelength λ instead of the frequency to characterize the colour of the intensity field.

5.1.2 Radiative Properties of Participating Media

Against a dark background, smoke is illuminated by rays from the sun falling on it obliquely from all directions except from behind; these rays are scattered by the smoke in every direction and some of the scattered rays enter our eyes and make the smoke visible. The particles which make up the smoke scatter blue light much more than red or yellow: therefore we see smoke as blue. On the other hand, when the background is bright, we see the smoke by transmitted light and it appears yellow because the blue in the incident white light has been scattered in all directions, very little can reach our eyes, and only the yellow and red remain to be transmitted and give colour to the smoke. *M. G. J. Minnaert*

In general, the radiative properties of a participating medium are a function of its density $\rho(\mathbf{x}, t)$ and temperature $T(\mathbf{x}, t)$. The evolution of these quantities for gaseous phenomena is introduced in the next chapter. In this section, we relate these quantities to the radiative properties of the medium. A participating medium modifies the propagation of light through three phenomena: emission, scattering and absorption. We review the modelling of these phenomena in the context of transport theory next. First, we assume that the time evolution of the quantities characterizing the medium is of several orders of magnitude slower than the speed of light. Therefore, the time dependence of these quantities will be ignored, i.e., $\rho(\mathbf{x}, t) \to \rho(\mathbf{x})$ and $T(\mathbf{x}, t) \to T(\mathbf{x})$. The emission of light by a medium in *local thermodynamic equilibrium* (LTE) is entirely determined by its local temperature $T(\mathbf{x})$ through black-body radiation $B_\lambda(T)$ [16]:

$$Q_\lambda(\mathbf{x}) = E_\lambda B_\lambda(T(\mathbf{x})) = E_\lambda \frac{2h}{\lambda^5 c} \left[\exp\left(\frac{hc}{\lambda k T} \right) - 1 \right]^{-1}, \tag{5.1}$$

where k is Stefan-Boltzmann's constant. The black-body radiation characterizes both the magnitude and the spectrum of the emission. The multiplicative factor E_λ is a material dependent function and characterizes the colour of the radiation. This factor, for example, permits the modelling of "blue flames" since pure black-body radiation is stronger in the red-yellow frequencies. The black body emission over all wavelengths is given by

$$\int_0^\infty B_\lambda(T)\, d\lambda = \frac{2\pi^5 k^4}{15 h^3 c^2} T^4 = \sigma T^4.$$

In other words, the total power emitted increases as the fourth power of the temperature. Thick media in which scattering events abound can be assumed to be in LTE [16]. In this case the energy absorbed is in balance with the emitted energy. For many phenomena this situation is not achieved and the LTE assumption is introduced as a matter of convenience. The scattering properties of a medium are entirely described by its *albedo* Ω_λ and its *phase function* $p_\lambda(\mathbf{s}, \mathbf{s}')$. The albedo gives the fraction of light that is scattered versus that which is absorbed. The phase function models the spherical distribution of scattered light. For example, scattering in clouds is predominantly in the forward direction. Usually, the phase function is normalized such that its integral over all directions is 4π. In most practical applications the phase function depends only on the angle between the two directions: $\mu = \mathbf{s} \cdot \mathbf{s}'$ (or in this case , the cosine og the angle). A simple model which is often used in computer graphics is the *Henyey-Greenstein* approximation [6, 26]:

$$p_\lambda(\mu) \approx \frac{1 - \bar{\mu}_\lambda^2}{(1 + \bar{\mu}_\lambda^2 - 2\bar{\mu}_\lambda \mu)^{3/2}}, \tag{5.2}$$

where $\bar{\mu}_\lambda = \frac{3}{2} \int_{-1}^{+1} \mu\ p_\lambda(\mu)\ d\mu$ is the first moment of the phase function. For negative $\bar{\mu}_\lambda$ the phase function favours back scattering over forward scattering. The converse is true for positive $\bar{\mu}_\lambda$. The total amount of light which is scattered into a particular direction at any moment is given by summing up all incoming light weighted by the phase function:

$$\mathcal{S}\{I_\lambda\}(\mathbf{x}, \mathbf{s}, t) = \frac{1}{4\pi} \int_{4\pi} p_\lambda(\mathbf{s} \cdot \mathbf{s}') I_\lambda(\mathbf{x}, \mathbf{s}', t)\ d\mathbf{s}'. \tag{5.3}$$

The frequency of interactions of the intensity field with the medium is given by

$$cK_t(\mathbf{x}, \lambda) = c\sigma_{t,\lambda}\rho(\mathbf{x}),$$

where ρ is the density of the medium and the *extinction cross section* σ_t is equal to the sum of the scattering and absorption cross-sections defined by

$$\sigma_s = \Omega \sigma_t \quad \text{and} \quad \sigma_a = (1 - \Omega)\sigma_t,$$

respectively. Therefore, these quantities are related to the absorption and scattering rates defined in Section 2.2.1: $K_a = \sigma_a \rho$ and $K_s = \sigma_s \rho$.

Given this description of the participating medium, we can now state the linear transport equation for the propagation of light (Equation 2.3 without external forces):

$$\frac{1}{c}\frac{\partial I_\lambda}{\partial t} + \mathbf{s}\cdot\nabla I_\lambda = -\rho\sigma_{t,\lambda}\left(I_\lambda - \Omega_\lambda \mathcal{S}\{I_\lambda\} - (1-\Omega_\lambda)Q_\lambda\right). \tag{5.4}$$

We repeat the phenomenological interpretation of this equation: the change in intensity along a certain direction is equal to a gain in intensity due to inscatter and emission minus losses caused by outscatter and absorption. In computer graphics, only the steady state solution is of interest since the viewing time (e.g., shutter speed of the virtual camera) is many orders of magnitude larger than the time it takes for the propagation of light to reach a steady state. Hence

$$\frac{1}{c}\frac{\partial I_\lambda}{\partial t} \approx 0,$$

and consequently $I_\lambda(\mathbf{x},\mathbf{s},t) \rightarrow I_\lambda(\mathbf{x},\mathbf{s})$. Because there is no coupling between different wavelengths, the explicit dependence of all functions on the wavelength will be dropped in the remainder of this chapter. The boundary conditions of this equation are given by the geometry and the reflectance properties of the surfaces in the environment. The reflection from a surface is characterized by its *bidirectional reflectance density function* φ_{brdf} (BRDF) giving the fraction of the incoming intensity I_{in} from a direction $\mathbf{s}'$ that is reflected into another direction $\mathbf{s}$:

$$I_{out}(\mathbf{x},\mathbf{s})\, d\mathbf{s} = \varphi_{brdf}(\mathbf{x},\mathbf{s},\mathbf{s}')I_{in}(\mathbf{x},\mathbf{s}')(\mathbf{n}\cdot\mathbf{s}')\, d\mathbf{s}',$$

where $\mathbf{n}$ is the normal to the surface at the point $\mathbf{x}$. The total outgoing intensity reflected into a particular direction is then obtained by integrating over all incoming directions:

$$I_{out}(\mathbf{x},\mathbf{s}) = \int_{2\pi} \varphi_{brdf}(\mathbf{x},\mathbf{s},\mathbf{s}')I_{in}(\mathbf{x},\mathbf{s}')(\mathbf{n}\cdot\mathbf{s}')\, d\mathbf{s}'.$$

This equation forms the basis of most global illumination algorithms in computer graphics. In the particular case when the BRDF is constant, corresponding to a perfectly diffuse surface, this relation becomes:

$$I_{out}(\mathbf{x},\mathbf{s}') = \varphi_{brdf}\int_{2\pi} I_{in}(\mathbf{x},\mathbf{s})(\mathbf{n}\cdot\mathbf{s}')\, d\mathbf{s}'.$$

5.1.3 Reduced Incident Intensity and Diffuse Intensity

In practice, it is convenient to separate the intensity into the sum of two functions: the *reduced incident intensity* I_{ri} and the *diffuse intensity* I_d [31]. The reduced incident intensity is that part of the intensity entering the participating medium which is attenuated by both scattering and absorption. The diffuse intensity, on the other hand, is created entirely within the medium through the phenomenon of scattering. Figure 5.1 illustrates the meaning of these two terms. More precisely, consider the ray $\mathbf{x}_u = \mathbf{x}_0 - u\ \mathbf{s}$ connecting a point $\mathbf{x}_b$ on one of the surfaces of the environment to a point $\mathbf{x}_0$ within the medium (again see Fig. 5.1). The reduced incident intensity is then the fraction of the intensity $I(\mathbf{x}_b,\mathbf{s},t)$ coming from the surface which is not scattered away or absorbed by the participating medium along the ray:

$$I_{ri}(\mathbf{x}_0,\mathbf{s}) = I_{out}(\mathbf{x}_b,\mathbf{s})\tau(\mathbf{x}_b,\mathbf{x}_a). \tag{5.5}$$

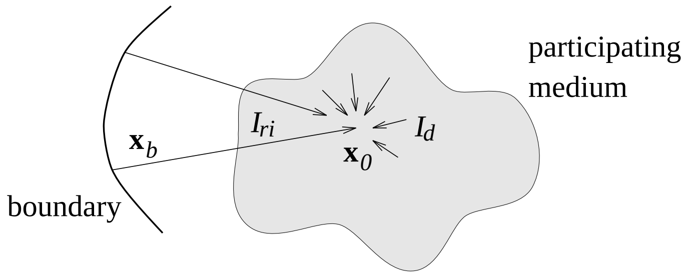

Figure 5.1: Reduced incident intensity I_{ri} vs diffuse intensity I_d.

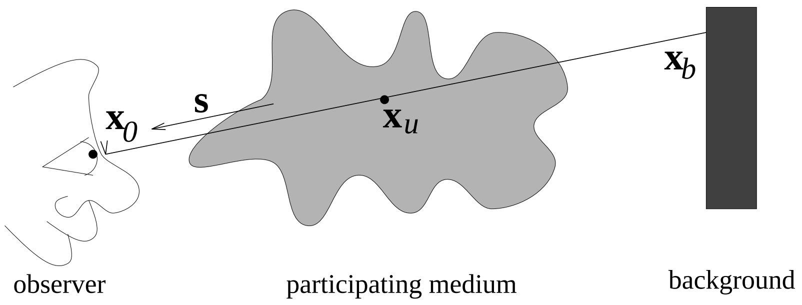

Figure 5.2: Integration of the transport equation along a viewing ray.

The *transparency* τ between two points $\mathbf{x}_b$ and $\mathbf{x}_0$ connected by a ray $\mathbf{x}_u$ is defined by:

$$\tau(\mathbf{x}_b, \mathbf{x}_0) = \exp\left(-\sigma_t \int_0^b \rho(\mathbf{x}_u)\, du\right). \tag{5.6}$$

Sometimes the *opacity* $\alpha = 1 - \tau$ is used instead. From the definition of the reduced incident intensity and the relation $I = I_{ri} + I_d$, it is possible to derive an equation for the diffuse intensity. In fact, this equation is identical to the transport equation (Eq. 5.4), with an additional source term J_{ri} due to the reduced incident intensity [31]:

$$J_{ri}(\mathbf{x}, \mathbf{s}) = \frac{\Omega}{4\pi} \int_{4\pi} p(\mathbf{s} \cdot \mathbf{s}') I_{ri}(\mathbf{x}, \mathbf{s}')\, d\mathbf{s}'. \tag{5.7}$$

The boundary condition for the diffuse intensity is that at a surface, the diffuse intensity cannot enter the medium: $I_d(\mathbf{x}, \mathbf{s}) = 0$ if $\mathbf{s}$ points into the medium. Note that the diffuse intensity represents the component of the intensity field which is difficult to compute. The reduced incident intensity, conversely, can be computed using standard integration procedures.

5.1.4 Integral Representation of the Transport Equation

The light reaching an observer at a point $\mathbf{x}_0$ from a direction $\mathbf{s}$ is given by integrating the transport equation along the ray connecting the background point $\mathbf{x}_b$ to the observer (see Figure 5.2). The result of this integration is [31]:

$$I(\mathbf{x}_0, \mathbf{s}) = I_{ri}(\mathbf{x}_0, \mathbf{s}) + I_d(\mathbf{x}_0, \mathbf{s}) = I_{out}(\mathbf{x}_b, \mathbf{s})\tau(\mathbf{x}_b, \mathbf{x}_0) + \int_0^b \tau(\mathbf{x}_u, \mathbf{x}_0)\sigma_t \rho(\mathbf{x}_u) J(\mathbf{x}_u, \mathbf{s})\, du, \tag{5.8}$$

where J is the *source intensity* and is the sum of the emitted intensity and the scattered intensity:

$$J(\mathbf{x},\mathbf{s}) = \Omega\mathcal{S}\{I(\mathbf{x},\mathbf{s})\} + (1-\Omega)Q(\mathbf{x}).$$

The scattered intensity can be further separated into a *diffuse source intensity* J_d and a component due to the first scatter of the reduced intensity:

$$\Omega\mathcal{S}\{I(\mathbf{x},\mathbf{s})\} = J_{ri}(\mathbf{x},\mathbf{s}) + J_d(\mathbf{x},\mathbf{s}), \tag{5.9}$$

where J_{ri} is defined by Equation 5.7. This separation is convenient in multiple scattering algorithms. Indeed we solve the diffuse source intensity using a diffusion approximation (see Section 5.5.2).

We can rewrite Equation 5.8 in terms of the transparency alone by defining the *average source function* along the ray by:

$$\begin{aligned}\bar{J}(\mathbf{x}_0,\mathbf{s}) &= \int_0^b \tau(\mathbf{x}_u,\mathbf{x}_0)\sigma_t\rho(\mathbf{x}_u)J(\mathbf{x}_u,\mathbf{s})\,du \Bigg/ \left(\int_0^b \tau(\mathbf{x}_u,\mathbf{x}_0)\sigma_t\rho(\mathbf{x}_u)\,du\right) \\ &= \frac{1}{1-\tau(\mathbf{x}_b,\mathbf{x}_0)}\int_0^b \tau(\mathbf{x}_u,\mathbf{x}_0)\sigma_t\rho(\mathbf{x}_u)J(\mathbf{x}_u,\mathbf{s})\,du. \end{aligned} \tag{5.10}$$

With this definition, the intensity of light reaching point $\mathbf{x}_0$ along the ray can be expressed as a linear combination of the background illumination $I_{out}(\mathbf{x}_b,\mathbf{s})$ and the average source function:

$$I(\mathbf{x}_0,\mathbf{s}) = \tau(\mathbf{x}_b,\mathbf{x}_0)I_{out}(\mathbf{x}_b,\mathbf{s}) + (1-\tau(\mathbf{x}_b,\mathbf{x}_0))\,\bar{J}(\mathbf{x}_0,\mathbf{s}). \tag{5.11}$$

We will use this formulation in our stochastic rendering algorithm (see Section 5.6).

5.1.5 Special Cases of the Integral Transport Equation

Usually, the integral transport equation is difficult to solve since the scattering term involves the intensity field. The integral on the right hand side of Equation 5.8 therefore cannot be integrated directly. However, there are some important special cases in which this integration is possible.

Total Emission

In highly emissive but tenuous or transparent gaseous phenomena such as fire, scattering effects can be ignored and hence $\Omega = 0$. In this particular case, the integral transport equation is an integration of the emission Q over the incident ray:

$$I(\mathbf{x}_0,\mathbf{s}) = \tau(\mathbf{x}_b,\mathbf{x}_0)I_{out}(\mathbf{x}_b,\mathbf{s}) + \int_0^b \tau(\mathbf{x}_u,\mathbf{x}_0)\sigma_t\rho(\mathbf{x}_u)Q(\mathbf{x}_u)\,du.$$

Specifically, when the emission is constant, $Q(\mathbf{x}) = Q_0$, we get an analytical expression for the intensity:

$$I(\mathbf{x}_0,\mathbf{s}) = \tau(\mathbf{x}_b,\mathbf{x}_0)I_{out}(\mathbf{x}_b,\mathbf{s}) + (1-\tau(\mathbf{x}_b,\mathbf{x}_0))Q_0. \tag{5.12}$$

Thus, the intensity depends solely on Q_0 and the transparency of the gaseous phenomenon.

Single Scattering Approximation

In media with low density or low albedo, scattering events are rare and the scattering term can be assumed to singularly depend on the reduced incident intensity. In this case, light reaches the observer from the light sources only after one scatter event. More precisely,

$$\mathcal{S}\{I\}(\mathbf{x},\mathbf{s}) = \mathcal{S}\{I_{ri} + I_d\}(\mathbf{x},\mathbf{s}) \approx \mathcal{S}\{I_{ri}\}(\mathbf{x},\mathbf{s}).$$

With these assumptions, the integral transport equation can be rewritten as:

$$\begin{aligned} I(\mathbf{x}_0,\mathbf{s}) &= \tau(\mathbf{x}_b,\mathbf{x}_0)I_{out}(\mathbf{x}_b,\mathbf{s}) + \\ & \int_0^b \tau(\mathbf{x}_u,\mathbf{x}_0)\sigma_t\rho(\mathbf{x}_u)\left(\Omega\mathcal{S}\{I_{ri}\}(\mathbf{x}_u,\mathbf{s}) + (1-\Omega)Q(\mathbf{x}_u)\right)\,du. \end{aligned} \tag{5.13}$$

This equation is known as the *single scattering integral transport equation*[1] and is used by most volume rendering algorithms in computer graphics. Blinn used this equation to obtain analytic solutions for constant planar density fields [5]. He applied his model to the rendering of the rings of Saturn. Klassen obtained similar analytic solutions to model the effects of the atmosphere and fog [38]. Max developed models for haze and simple clouds [54, 53]. By modelling the clouds as constant density fields bounded by a height field above and beneath a plane, and by assuming that the sun is at the zenith, he derived a simple solution. He simplified the model further by approximating the exponential function by a quadratic. Nishita et al. developed a similar model to render constant density maps bounded by convex polyhedra [63]. Shafts of light and shadowing effects were modelled using directional light sources and shadow volumes. In the absence of any assumptions about the geometry of the medium, the single scattering scattering equation is solved by direct numerical integration along rays cast from the viewer into the medium [35, 84, 30]. In this situation, the medium is typically given as a set of voxels. Hierarchical representations of the medium were used to speed up ray casting and to reduce aliasing effects [48, 86]. Volume coherence can be exploited even more by using scan-line algorithms to render the volume. These methods either process the voxel data base from back to front [15] or from front to back [98] using the techniques of digital compositing [76]. Ebert et al. designed an efficient algorithm by modifying a A-buffer scanliner [17].

Typically, though, the contribution of the diffuse intensity to the scattering term is non-constant and has to be calculated using more sophisticated techniques. In Section 5.5.2 we review work done in this area and present a new algorithm based on a diffusion process.

5.2 Representation of the Intensity Field

We now introduce a representation for the intensity field that will permit us to compute an approximation to the general solution of Equation 5.8. We model the spatial structure of the intensity field using a blob representation (Equation 4.8):

$$I_\sigma(\mathbf{x},\mathbf{s}) = \frac{1}{\rho_\sigma(\mathbf{x})}\sum_{k=1}^{N} m_k I_k(\mathbf{s}) W(\mathbf{x}-\mathbf{x}_k). \tag{5.14}$$

[1]To account for the effects of multiple scattering, a factor $(1-\tau(\mathbf{x}_b,\mathbf{x}_0))I_d^0$, where I_d^0 is a constant, can be added to the right hand side of this equation. This term is useful in practical simulations in order to "brighten" up pictures of gases.

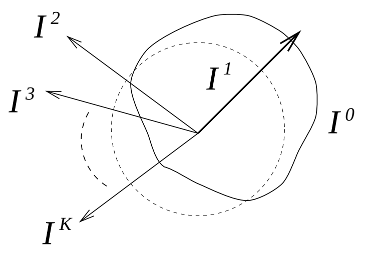

Figure 5.3: The angular distribution of the source intensity.

where each coefficient in the expansion depends on the angular variable. At this point, we could expand each of these coefficients into an orthonormal basis of functions defined on the unit sphere. The standard polynomials with this property are the *spherical harmonics* (see Appendix 5.A of this chapter). To approximate highly spiked distributions, many harmonics of a high degree are required. Unfortunately, in this situation, the number of harmonic basis functions of a given degree grows quadratically. In practice, then, spherical harmonics are too expensive to represent arbitrary distributions. In order to resolve this problem, we split the dependence on the angular variable into a diffuse component I_{diff} and a specular component I_{spec}. This separation is common in computer graphics surface reflectance models [75]. The diffuse component is given by an expansion of the intensity field into a fixed number of spherical harmonics. The first term in the expansion corresponds to the *average intensity* over all angles:

$$I^0(\mathbf{x}) = \frac{1}{4\pi} \int_{4\pi} I(\mathbf{x}, \mathbf{s})\, d\mathbf{s}.$$

The next three terms in the harmonic expansion can be combined into a single *average intensity flux* giving the principal direction of the flux of intensity:

$$\mathbf{I}^1(\mathbf{x}) = \frac{3}{4\pi} \int_{4\pi} I(\mathbf{x}, \mathbf{s})\, \mathbf{s}\, d\mathbf{s}.$$

We assume in the remainder of this thesis that these two functions are sufficient to characterize the diffuse component of the intensity field:

$$I_{diff}(\mathbf{x}, \mathbf{s}) = I^0(\mathbf{x}) + \mathbf{I}^1(\mathbf{x}) \cdot \mathbf{s}.$$

Our presentation, however, can be extended to a higher number of harmonics. We will come back to this point and discuss other possible extensions of the algorithm in Section 5.4.8. The specular component of the intensity field is modelled as a sum of impulses centred at the directions $\mathbf{s}_2, \cdots, \mathbf{s}_K$:

$$I_{spec}(\mathbf{x}, \mathbf{s}) = \sum_{l=2}^{K} I^l(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}_l).$$

For example, one of these impulses could model the scattering of the light coming from the sun. See Figure 5.3 for an illustration of the shape of the angular distribution. We can use this expansion to complete the description of the intensity field. Indeed, each spatial coefficient can in turn be expanded in its angular variable:

$$I_k(\mathbf{s}) = I_k^0 + \mathbf{I}_k^1 \cdot \mathbf{s} + \sum_{l=2}^{K_k} I_k^l \delta(\mathbf{s} - \mathbf{s}_l). \tag{5.15}$$

The intensity consequently is represented by a total of $(K+1)N$ coefficients.

5.2.1 Representation of the Radiative Properties

The scattering properties of the participating medium are characterized by its phase function $p(\mathbf{s}\cdot\mathbf{s}')$. Similar to the angular representation of the intensity, we separate the phase function into a diffuse component and a specular component. The equivalent of the average intensity and the average flux of the intensity field for the phase function are the first two moments defined by:[2]

$$p^0 = \frac{1}{2}\int_{-1}^{1} p(\mu)\,d\mu = 1 \quad \text{and} \quad p^1 = \frac{3}{2}\int_{-1}^{1} \mu\, p(\mu)\,d\mu = \bar{\mu}.$$

The specularity of most phase functions encountered in practice are characterized either by strong forward scattering or by strong backward scattering. Instead of considering any number of discrete directions, we model the specularity of the phase function by a single term. Following these considerations, we represent the phase function by

$$p(\mathbf{s}\cdot\mathbf{s}') = 1 + \bar{\mu}\;\mathbf{s}\cdot\mathbf{s}' + p^2\delta(\mathbf{s}\cdot\mathbf{s}' - \mu_2), \tag{5.16}$$

where μ_2 is either equal to $+1$ (forward scattering) or equal to -1 (backward scattering).

The emission function does not depend on direction and is therefore expanded into a blob representation only:

$$Q_\sigma(\mathbf{x}) = \frac{1}{\rho_\sigma(\mathbf{x})}\sum_{k=1}^{N} m_k Q_k W(\mathbf{x}-\mathbf{x}_k),$$

where each coefficient is entirely determined by the temperature of the gas: $(Q_\lambda)_k = E_\lambda B_\lambda(T(\mathbf{x}_k))$ (see Equation 5.1). The source intensity is thus represented as:

$$J_\sigma(\mathbf{x},\mathbf{s}) = \Omega\mathcal{S}\{I_{ri,\sigma}\}(\mathbf{x},\mathbf{s}) + \Omega\mathcal{S}\{I_{d,\sigma}\}(\mathbf{x},\mathbf{s}) + (1-\Omega)Q_\sigma(\mathbf{x}),$$

where both the reduced incident intensity and the diffuse intensity are expanded into the representation given in the previous subsection.

5.3 Blob Integration

The puzzle of perspective representation is that it make things look right by doing them wrong. *Rudolf Arnheim*

In Section 5.1, we have encountered many equations which necessitate the integration of a blob representation over a given ray. This is the case for the transparency τ which involves an integral over the density field, for both the total emission approximation and the single scattering approximation. To integrate these functions, we could use a standard integration schemes. This technique, however, produces visible aliasing artifacts, unless a large number of samples is used. These artifacts are visible especially in animations. The motivation behind integration algorithms is to chose the samples according to the blob representation. This has two advantages. Firstly, since we know the shape of the function, we can limit the number of samples by choosing them at the "right place". Secondly, in an animation the samples move

[2]See the addition theorem in Appendix 5.A of this chapter

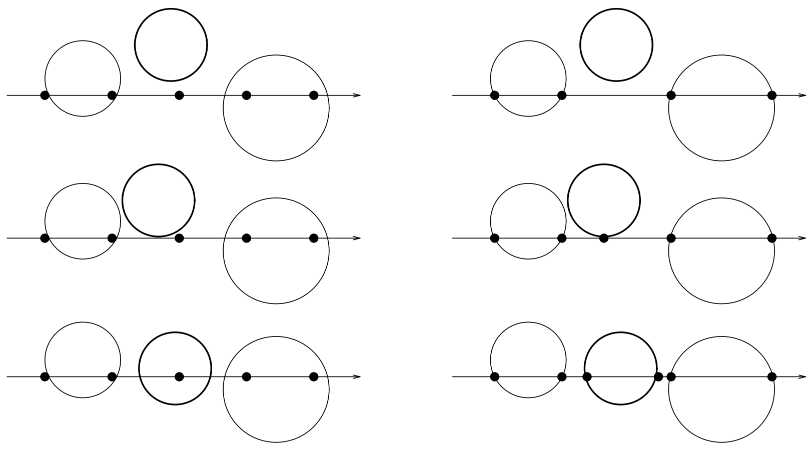

Figure 5.4: Uniform sampling (left) versus blob sampling (right). In the blob sampling the samples are placed according to the position of the blobs. Sudden discontinuities in animations due to moving blobs are handled without over sampling.

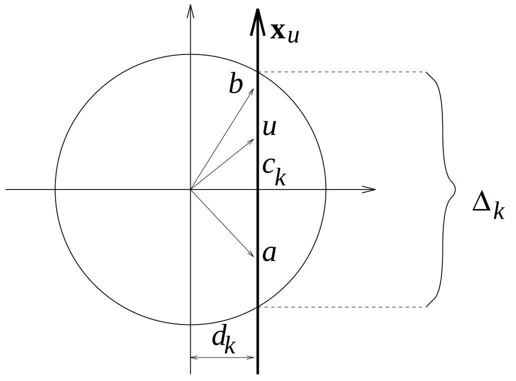

Figure 5.5: Geometry of the ray intersecting a truncated blob.

naturally with the blob representation, reducing the occurrence of visible artifacts over time. This is illustrated in Figure 5.4. For example, if a single blob is moving,the regular sampling technique misses the blob when it comes into contact with the ray. The blob is added to the integral only when there is considerable overlap. This discontinuity will produce flickering in an animation. The blob sampling method, conversely, detects the blob right from the beginning. Hence, we are mainly interested in choosing integration schemes which produce consistent depictions of the phenomenon. High numerical precision is not necessary. Indeed, our blob representation is, in the first place, an approximation. In the remainder of this section we assume that a hierarchical data structure of bounding spheres has been computed for the truncated blobs using the algorithm of Section 4.2.2.

5.3.1 Transparency

The transparency along a given ray is a function of the integral of the density field over the ray (see Equation 5.6). Using the blob representation for the density, this integral for an arbitrary segment $[a, b]$ on the ray reduces to:

$$\Gamma(b,a) = \sum_{k=1}^{N} m_k \int_a^b W(\mathbf{x}_u - \mathbf{x}_k, \sigma)\, du = \sum_{k=1}^{N} m_k \Gamma_k(b,a). \tag{5.17}$$

We develop algorithms for smoothing kernels depending only on the distance between two points. From the geometry of Figure 5.5, we consequently can deduce that the integral Γ_k depends both on the distance d_k of the ray to the blob and on the point on the ray c_k closest to the centre of the blob:

$$|\mathbf{x}_u - \mathbf{x}_k| = \sqrt{d_k^2 + (u - c_k)^2},$$

In addition, each integral depends on the smoothing length σ and on the bounds of the integral a and b. In principle, the integral can then be computed by table lookup, by precomputing the following integrals and storing them in a table:

$$\Gamma_k(b, 0) = \int_0^b W\left(\sqrt{d_k^2 + (u - c_k)^2}, \sigma\right)\, du.$$

The integrals in Equation 5.17 can thus be computed from this table via

$$\Gamma_k(b, a) = \Gamma_k(b, 0) - \Gamma_k(a, 0).$$

However, the table is four-dimensional and becomes too large when high precision is required. For Gaussian smoothing kernels, some simplifications can be made. This follows from the fact that the dependence on the distance d_k can be taken out of the integral:

$$\exp\left(-\frac{d_k^2 + (u - c_k)^2}{2\sigma^2}\right) = \exp\left(-\frac{d_k^2}{2\sigma^2}\right)\exp\left(-\frac{(u - c_k)^2}{2\sigma^2}\right).$$

By using the substitution $v = (u - c_k)/\sqrt{2}\sigma$ in the integral we obtain:

$$\begin{aligned}\Gamma_k(b, a) &= \frac{1}{(2\pi)^{3/2}\sigma^3}\exp\left(-\frac{d_k^2}{2\sigma^2}\right)\sqrt{2}\sigma\int_{\frac{a-c_k}{\sqrt{2}\sigma}}^{\frac{b-c_k}{\sqrt{2}\sigma}}\exp\left(-v^2\right)\, dv \\ &= \frac{1}{4\pi\sigma^2}\exp\left(-\frac{d_k^2}{2\sigma^2}\right)\left(\operatorname{erf}\left(\frac{b - c_k}{\sqrt{2}\sigma}\right) - \operatorname{erf}\left(\frac{a - c_k}{\sqrt{2}\sigma}\right)\right),\end{aligned}$$

where “erf” is the *error function* defined by:

$$\operatorname{erf}(v) = \frac{2}{\sqrt{\pi}}\int_0^v \exp\left(-w^2\right)\, dw.$$

Both the exponential and the error function are available on most systems and are usually optimized through a table lookup scheme.

Although the derivation just presented gives an accurate evaluation of the integral, we have found that a simple linear approximation of the erf function produces results which are visually indistinguishable from the exact method. We approximate the erf function within the interval of overlap with the truncated Gaussian using the following piece-wise linear function:

$$\operatorname{lerf}(v) = \begin{cases} -1 & \text{if} \quad v < -1/2 \\ 2v & \text{if} \quad |v| \le 1/2 \\ 1 & \text{if} \quad v > 1/2 \end{cases}$$

Figure 5.6 shows this approximation along with the error function. If we assume that both a and b lie within the truncated blob[3], then the difference

$$\operatorname{lerf}\left(\frac{b - c_k}{\Delta_k}\right) - \operatorname{lerf}\left(\frac{a - c_k}{\Delta_k}\right) = \frac{b - a}{\Delta_k}.$$

[3]This situation can always be achieved by “clipping” the values of a and b to the truncated interval.

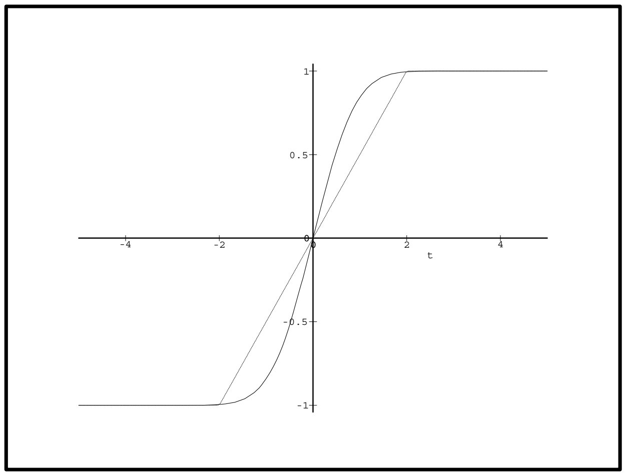

Figure 5.6: Error function and the linear approximation that we use.

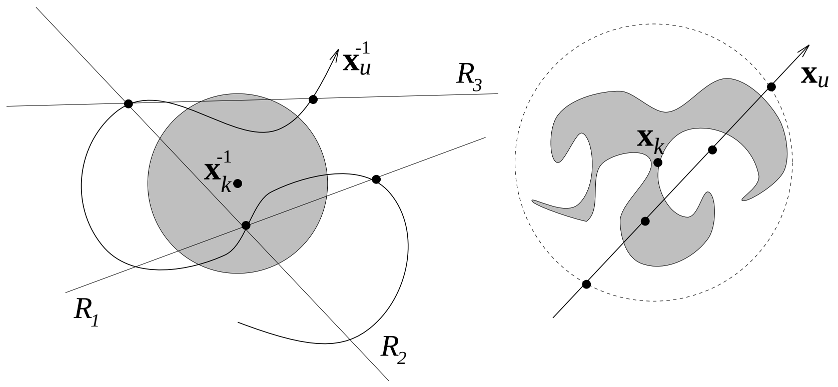

Figure 5.7: Integrating a warped blob. Instead of warping the blob we backtrace several samples on the ray through the wind field.

We rewrite the integral Γ_k in terms of the linear approximation:

$$\Gamma_k(b,a) \approx \frac{1}{(2)^{3/2}\pi\sigma^3} \exp\left(-\frac{d_k^2}{2\sigma}\right) \frac{b-a}{\Delta_k} = \frac{b-a}{\Delta_k}\sqrt{\pi}G_\sigma(d_k),$$

where Δ_k is the length of the region of overlap of the truncated blob with the ray (see Figure 5.5). Usually, we can apply the same approximation to any smoothing kernel which has a "Gaussian-like shape":

$$\Gamma_k(b,a) \approx \frac{b-a}{\Delta_k} W(d_k,\sigma). \tag{5.18}$$

The transparency between two points is then given by summing up the contribution of each blob and taking an inverse exponential.

This technique works for regular spherical blobs. To handle warped blobs the algorithm has to be modified slightly, however. There are two methods of doing this. We can either integrate the warped blob along the ray or we can backwarp the ray and integrate the unwarped blob over a curve. Figure 5.7 illustrates the difference between the two approaches. We use the former one because it takes us back to the spherical blobs integration algorithm. We sample the interval of overlap of $[a,b]$ with the sphere bounding the warped blob on a finite number M of samples u_l. We then backtrace the points $\mathbf{x}_{u_l}$ corresponding to these

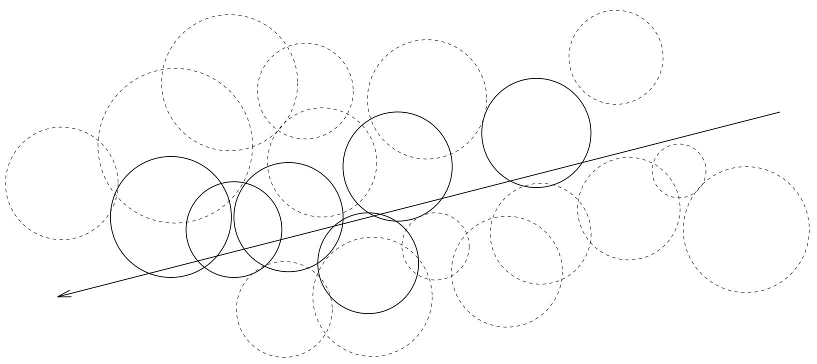

Figure 5.8: Selecting the truncated blobs that intersect the ray.

samples through the force field yielding the samples $\mathbf{x}_{u_l}^{-1}$ on the backwarped ray. recalling Figure 5.7, each pair of backwarped points $(\mathbf{x}_{u_l}^{-1}, \mathbf{x}_{u_{l+1}}^{-1})$ defines a ray R_l. As in the unwarped case, we can calculate both the overlap $\Delta_{k,l}$ and the distance $d_{k,l}$ to the centre of the unwarped blob for each of these rays. The transparency along each ray is then approximated using Equation 5.18. By adding up the contributions from each of the $M-1$ rays, we obtain an approximation of the value of the total integral:

$$\Gamma(b,a) \approx \sum_{l=1}^{M-1} \frac{b-a}{\Delta_{k,l}} W(d_{k,j}, \sigma).$$

The additional cost of the warping is thus linear in the number of samples chosen and is independent of the number of blobs.

5.3.2 Integral Transport Equation

We extend these integration techniques to the evaluation of the integral appearing in the integral transport equation (Equation 5.8). We assume in this section that the coefficients appearing in the representation of the source intensity J have been computed, e.g., by the shooting algorithm presented in the next section. We will not derive exact expressions for the integrals but rather use the approximation given in Equation 5.18 when applicable. Prior to integrating, we determine the blobs whose truncated domains intersect the ray and store them into a list L (see Figure 5.8). This is done using the hierarchical representation of the blobs by the following recursive algorithm:

```
SelectBlobs($b$)
    if $b \rightarrow$left$= 0$ and $b \rightarrow$right$= 0$ then    { this is a leaf blob }
        Compute distance to ray $d_k$
        if $d_k <$ radius of blob then
            $L = L \cup \{b\}$                       { add blob to list }
        end if
    else
        SelectBlobs($b \rightarrow$left)
        SelectBlobs($b \rightarrow$right)
    end if
```

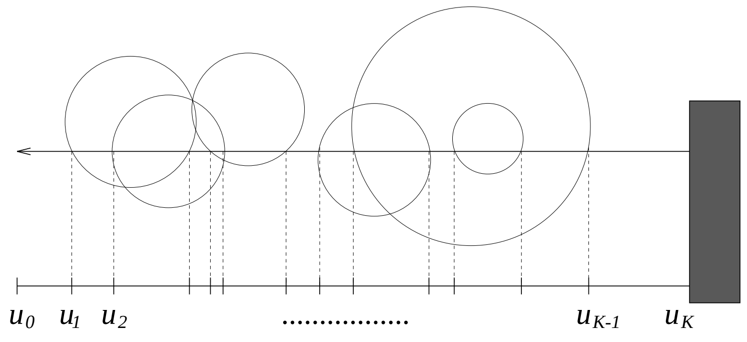

Figure 5.9: Subdivision of the ray induced by the blob overlap.

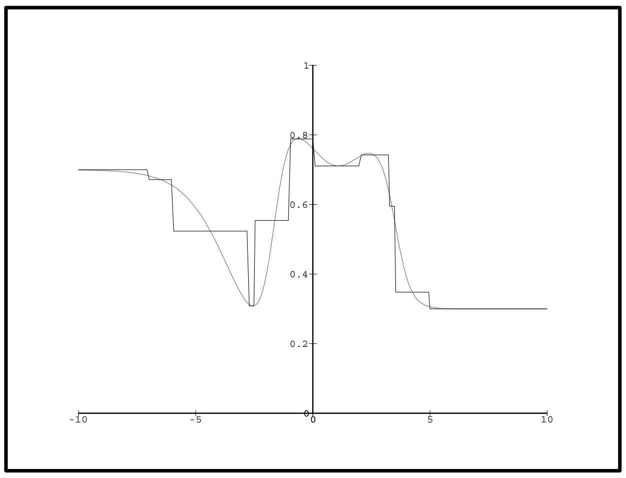

Figure 5.10: The source intensity is approximated by a piecewise constant function.

We obtain the list by calling the routine with the root of the blob tree as its argument. When the blobs are distributed fairly uniformly in space, each ray intersect on average $N^{1/3}$ blobs, and the cost to reach these blobs is $O(N^{\frac{1}{3}} \log N)$. Once the list is constructed, we subdivide the ray into K_i disjoint intervals $[u_j, u_{j+1}]$ induced by the overlap of the blobs as shown in Figure 5.9. The overlap of each blob is denoted again by Δ_k. We approximate the source intensity J on each of these intervals by its value at the midpoint $v_j = (u_j + u_{j+1})/2$ of the interval:

$$J(\mathbf{x}_u, \mathbf{s}) \approx J(\mathbf{x}_{v_j}, \mathbf{s}) = \frac{\sum_k m_k J_k(\mathbf{s}) W(\mathbf{x}_{v_j} - \mathbf{x}_k)}{\sum_k m_k W(\mathbf{x}_{v_j} - \mathbf{x}_k)}, \tag{5.19}$$

where the sums are over the blobs overlapping the interval, i.e., $k \in L \cap [u_j, u_{j+1}]$. This approximation corresponds to the piecewise constant function depicted in Figure 5.10. With this approximation, the integral of Equation 5.8 can be integrated exactly on each interval:

$$\int_{u_j}^{u_{j+1}} \tau(\mathbf{x}_u, \mathbf{x}_0)\sigma_t \rho(\mathbf{x}_u) J(\mathbf{x}_u, \mathbf{s})\, du \approx J(\mathbf{x}_{v_j}, \mathbf{s})\tau(\mathbf{x}_{u_j}, \mathbf{x}_0)(1 - \tau(\mathbf{x}_{u_{j+1}}, \mathbf{x}_{u_j})),$$

for $j = 0, \cdots, K_i - 1$. The transparency of each interval $\tau_j = \tau(\mathbf{x}_{u_{j+1}}, \mathbf{x}_{u_j})$ can be computed using the approximation given in the previous section (Equation 5.18). By adding up the integrals for each interval, the integral in Equation 5.8 is approximated by:

$$I_d(\mathbf{x}_0, \mathbf{s}) \approx \sum_{j=1}^{K_i} \left(\prod_{j'<j} \tau_{j'} \right) (1 - \tau_j) J(\mathbf{x}_{v_j}, \mathbf{s}).$$

We have replaced the integral by a finite sum and product over the intervals. We now present an algorithm which evaluates this sum. Observe that the product appearing in the sum is equal to the total transparency up to the j-th interval decreases with j. The summation can consequently be terminated as soon as this transparency falls below a certain threshold, if an upper bound on the $J(\mathbf{x}_{v_j}, \mathbf{s})$ is known.

```
{ Initialize variables }
I_d = 0
tau_tot = 1
{ now render from front to back }
for $j = 0, \cdots, K-1$ do
    rho = T = J = 0
    for each blob $k \in L$ which overlaps $[u_j, u_{j+1}]$ do
        m = $m_k W(\mathbf{x}_v - \mathbf{x}_k)$
        rho = rho + m
        J = J + m*$J_k(\mathbf{s})$
        T = T + $\frac{u_{j+1} - u_{u_j}}{\Delta_k} W(d_k, \sigma)$
    end for
    tau = exp(-$\sigma_t$*T)
    I_d = I_d + tau_tot*(1-tau)*J/rho
    tau_tot = tau_tot*tau
    if tau_tot < EPS then exit loop
end for
{ combine with background intensity obtained using a standard ray tracer }
```

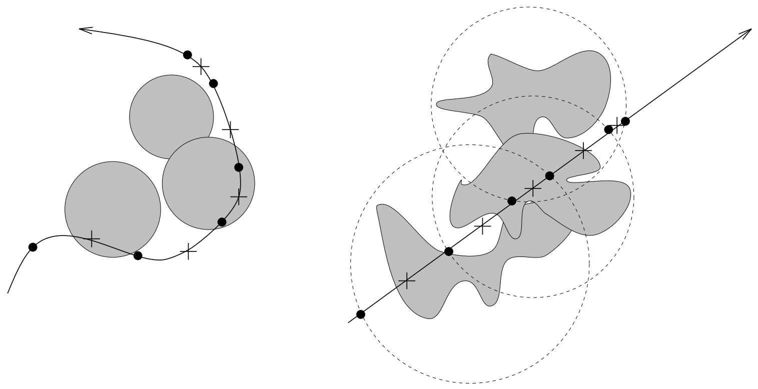

Figure 5.11: Each interval is backwarped through the motion field.

```
I = tau_tot*$I_{out}(\mathbf{x}_b, s)$ + I_d
```

The number of intervals is at most twice the number $|L|$ of blobs intersecting the ray. Both loops in this algorithm are therefore of $O(|L|)$ and the the total complexity is $O(|L|^2)$. When there is a large overlap of blobs, this algorithm becomes expensive. In Appendix 5.C of this chapter we give an $O(|L|)$ algorithm obtained by an additional approximation of the source term.

The extension of the above algorithm to warped blobs is straightforward. For each interval $[u_j, u_{j+1}]$, we backwarp the midpoint v_j to obtain $\mathbf{x}_{v_j}^{-1}$ which we then use to evaluate the smoothing kernel

$$W(\mathbf{x}_{v_j} - \mathbf{x}_k, \sigma) \rightarrow W\left(\mathbf{x}_{v_j}^{-1} - \mathbf{x}_k^{-1}, \sigma\right).$$

The procedure is shown in Figure 5.11 where the midpoints are denoted by a "+".

5.4 A Shooting Algorithm for Density Fields

The destruction of local colour had been carried to the extreme by the impressionists, who had used reflections to apply the green of meadow to the body of a cow or the blue of the sky to the stones of a cathedral. In consequence, modern artists became free not only to make a red object blue, but also to replace the unity of one local colour with any combination of different colours. *Rudolf Arnheim*

In this section we present a method of computing an approximation to the solution of the transport equation. The algorithm is an extension of the progressive refinement method from radiosity to environments containing participating media. This algorithm was first developed for environments composed of opaque diffuse surfaces only [14]. The surfaces of the environment first are subdivided into patches. At each step of the algorithm the energy from the brightest patch is *shot* into the environment and collected at the other patches. As the algorithm proceeds, the environment "brightens up" and successive approximations to the solution of the transport equation are computed. The method has been extended to surfaces having more general reflectance properties using spherical harmonics [92]. We extend the shooting algorithm to environments containing participating media modelled by a blob representation. We assume that the surfaces in the environment are opaque diffuse

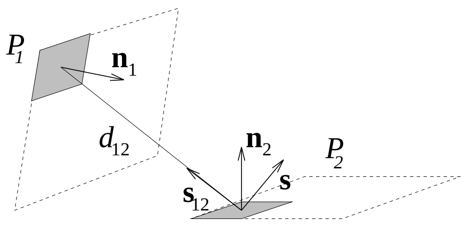

Figure 5.12: Form factor computation between two patches.

reflectors with a constant emission. In Section 5.4.8 we mention how our algorithm can be extended to surfaces having more general reflectances. We also include non-physical light sources, such as point lights, in our formulation. These light sources are used in most computer graphics illumination models and are good approximations to actual light sources. For example, the directional light source is a good approximation of sunlight on a bright day. A general description of our algorithm is discussed next.

```
set intensity and unshot intensity of blobs and patches to their emission
shoot from non-physical light sources to all blobs/patches
while unshot energy is below a threshold do
    choose blob/patch with highest unshot power
    if patch then
        shoot power to all other patches
        shoot power to all blobs
    end if
    if blob then
        shoot power to all patches
        shoot power to other blobs
    end if
end while
```

Consequently, we must develop shooting algorithms for the three shooting operations involving a blob: patch $\rightarrow$ blob, blob $\rightarrow$ patch and blob $\rightarrow$ blob. These operations and the patch to patch shooting operations are described in what follows.

5.4.1 Patch to Patch

The intensity of light leaving a patch P_1 and reaching a patch P_2 along a direction $\mathbf{s}_{12}$ is given by:

$$dI_{12}^{in}(\mathbf{s}) = I_1\,(\mathbf{n}_2 \cdot \mathbf{s}_{12})\,d\mathbf{s},$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ denote the normals of patch P_1 and patch P_2, respectively. This equation can be written in terms of an infinitesimal area dA_1 by considering the solid angle subtended by the area:

$$d\mathbf{s} = (-\mathbf{n}_1 \cdot \mathbf{s}_{12})\frac{dA_1}{d^2},$$

where d is the distance between the points on the two patches. The total intensity incident at a point on patch P_2 due to a patch P_1 is then given by integrating over the whole area

A_1 of patch P_1:

$$I_{12}^{in} = I_1 \int_{A_1} dI_{12}(\mathbf{s}) = I_1 \int_{A_1} \frac{(-\mathbf{n}_1 \cdot \mathbf{s})(\mathbf{n}_2 \cdot \mathbf{s})}{d^2}\, dA_1 = I_1 F_{12}.$$

The integral F_{12} is called a *form factor* and, in the case of diffuse surfaces, depends solely on the geometry of the patches. The form factor in general can be computed numerically by transforming the area integral into a contour integral [2]. For simple geometries, analytic expressions exist for the form factor. The form factor between a disk, of radius r with a normal $\mathbf{n}_1$, and a point on a patch of normal $\mathbf{n}_2$ is given by [90]:

$$F_{12}^{disk} = \frac{r^2}{r^2 + d_{12}^2}(\mathbf{n}_1 \cdot \mathbf{s}_{12})(-\mathbf{n}_2 \cdot \mathbf{s}_{12}),$$

where d_{12} is the distance between the centre of the disk and the point on the patch and $\mathbf{s}_{12}$ is its direction. When the shooting patch is small or the distance d_{12} is large, we can approximate it by a disk with equivalent area and use this form factor to compute the intensity incident on patch P_2:

$$I_{12}^{in} = I_1 F_{12}^{disk}.$$

The BRDF φ_2 of patch P_2 then gives the fraction of this light which is reflected:

$$I_{12}^{out} = \varphi_2 I_{12}^{in} = \varphi_2 I_1 F_{12}^{disk}.$$

Arbitrary patches can be broken up into smaller elements, which are approximately disk shaped and the intensity can be shot from each element [101].

5.4.2 Patch to Blob

The shooting operation from a patch to a blob is actually a special case of the patch to patch shooting operation. In fact, consider a small surface element at the centre of the blob whose normal is aligned with the direction $\mathbf{s}_{12}$, then $\mathbf{n}_2 \cdot \mathbf{s}_{12} = 1$ and the incident intensity on the blob is

$$J_{12}^{in} = I_1 F_{12}^{disk} = I_1 \frac{r^2}{r^2 + d_{12}^2}(-\mathbf{n}_2 \cdot \mathbf{s}_{12}).$$

When this light reaches the blob, a fraction $1 - \Omega$ of it is absorbed and a fraction Ω of it is scattered into other directions. The scattered intensity due to the shooting patch is therefore

$$J_{12}^{out}(\mathbf{s}) = I_1 F_{12}^{disk} \Omega p(\mathbf{s}_{12} \cdot \mathbf{s}).$$

Using the angular representation of the phase function, this relation is actually

$$J_{12}^{out}(\mathbf{s}) = I_1 F_{12}^{disk} \Omega \left(1 + \bar{\mu}\ \mathbf{s}_{12} \cdot \mathbf{s} + p^2 \delta(\mathbf{s}_{12} \cdot \mathbf{s} - \mu_2)\right).$$

We emphasize that this is an equation for the coefficients appearing in the angular expansion of the source intensity. This equation states how these coefficients should be updated at a shooting step, i.e.,

$$J_2^0 \longrightarrow J_2^0 + I_1 F_{12}^{disk} \Omega \quad \text{and} \quad \mathbf{J}_2^1 \longrightarrow \mathbf{J}_2^1 + \bar{\mu}\ I_1 F_{12}^{disk} \Omega\ \mathbf{s}_{12}.$$

In case the coefficient p^2 is non-zero, an additional direction has to be added to the specular component of the outgoing intensity:

$$J_2^{K_2+1} = p^2 I_1 F_{12}^{disk} \Omega\ \mu_2 \mathbf{s}_{12} \quad \text{and} \quad K_2 \longrightarrow K_2 + 1.$$

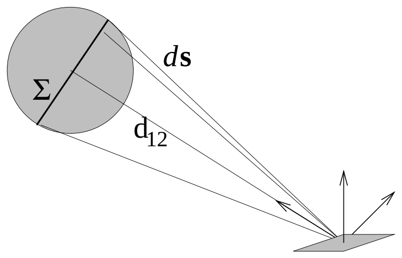

Figure 5.13: Form factor between a blob and a patch.

5.4.3 Blob to Patch

To derive a form factor between a blob and a patch we assume that the distant separating them is large enough. In this case the distance between the patch and any point in the blob is approximately equal and parallel to the distance d_{12} between the centre of the patch and the centre of the blob. If the patch is assumed to be aligned with this distance, the form factor is given by the integral over the cross section Σ normal to d_{12} (see Figure 5.13):

$$F_{12} = \int_\Sigma \left(\int_{-\infty}^{\infty} W(x,y,z,\sigma)\, dz \right) d\mathbf{s} = \int_\Sigma \left(\int_{-\infty}^{\infty} W(x,y,z,\sigma)\, dz \right) \frac{dx\ dy}{d_{12}^2} = \frac{1}{d_{12}},$$

where the last equality follows from the fact that the smoothing kernel is normalized. A similar result was obtained for the form factor between a cube of volume and a patch [83]. The shooting operation of a blob to a patch is then a particular case of the patch to patch shooting operation with $-\mathbf{n}_1 \cdot \mathbf{s}_{12} = 1$. Hence, the reflected intensity at the patch caused by the intensity coming from the blob is given by:

$$I_{12}^{out} = \varphi_2 F_{12} J_1(\mathbf{s}_{12}) = \varphi_2 F_{12} \left(J_1^0 + \mathbf{J}_1^1 \cdot \mathbf{s}_{12} + \sum_{l=2}^{K_1} J_1^l \delta(\mathbf{s}_{12} - \mathbf{s}_l) \right).$$

5.4.4 Blob to Blob

The shooting operation between two blobs is a particular case of the blob to patch shooting operation, where the receiving patch is aligned with the direction, hence $-\mathbf{n}_1 \cdot \mathbf{s}_{12} = 1$ and $\mathbf{n}_2 \cdot \mathbf{s}_{12} = 1$. In this case, the scattered intensity is given by

$$J_{12}^{out}(\mathbf{s}) = \Omega p(\mathbf{s}_{12} \cdot \mathbf{s}) J_1(\mathbf{s}_{12}) F_{12}.$$

As with the patch to blob operation, each coefficient in the angular representation of the receiving blob has to be updated.

5.4.5 Light Sources

We derive the shooting operations from three type of light sources commonly used in computer graphics: directional light sources, point light sources and spotlights. A directional light source $I_{dir}(\mathbf{s})$ is defined by a unique direction $\mathbf{s}_{dir}$ and an intensity I_{dir}^0:

$$I_{dir}(\mathbf{s}) = I_{dir}^0 \delta(\mathbf{s} - \mathbf{s}_{dir}).$$

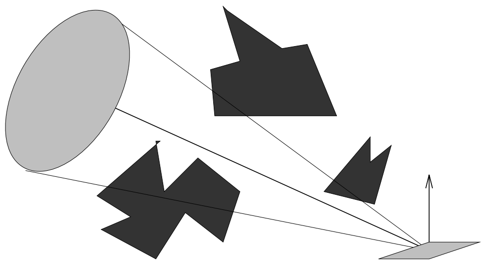
Figure 5.14: The shadow is computed by tracing shadow rays.

The amount of directional light which is reflected by a patch P_2 with a normal $\mathbf{n}_2$ is given by:

$$I^{out}_{1,2} = \varphi_2 I^0_{dir}(-\mathbf{n}_2 \cdot \mathbf{s}_{dir}).$$

Similarly, the amount of directional light scattered by a blob B_2 is given by

$$J^{ou}_{12}(\mathbf{s}) = \Omega p(\mathbf{s} \cdot \mathbf{s}_{dir}) I^0_{dir} = I^0_{dir}\left(1 + \bar{\mu}\ \mathbf{s}_{dir} \cdot \mathbf{s} + p^2 \mu_2\ \delta(\mathbf{s} \cdot \mathbf{s}_{dir} - \mu_2)\right).$$

A point light source is a special case of a shooting step from an infinitesimal blob:

$$F_{12} = \frac{1}{d^2_{12}}.$$

The intensity of the reflected light at a patch P_2 is therefore:

$$I^{out}_{12} = \varphi_2 I^0_{pnt} \frac{\mathbf{n}_2 \cdot \mathbf{s}_{12}}{d^2_{12}}.$$

Likewise for a blob:

$$J^{out}_{12}(\mathbf{s}) = p(\mathbf{s}_{12} \cdot \mathbf{s}) I^0_{pnt} \frac{1}{d^2_{12}} = \frac{I^0_{pnt}}{d^2_{12}}\left(1 + \bar{\mu}\ \mathbf{s}_{12} \cdot \mathbf{s} + p^2 \delta(\mathbf{s}_{12} \cdot \mathbf{s} - \mu_2)\right).$$

A spot light is a point light source with a direction $\mathbf{s}_{sl}$ specifying concentration of light. The spot light intensity at the location of its source has the following distribution in general:

$$I_{sl}(\mathbf{s}) = I^0_{sl} f(\mathbf{s}_{sl} \cdot \mathbf{s}),$$

where f is an arbitrary function modelling the spread and shape of the distribution of light around the central direction. The form factor is then identical to that of a point source. The shooting equations are also similar, only the intensity I^0_{pnt} has to be replaced with $I^0_{sl} f(\mathbf{s}_{12} \cdot \mathbf{s}_{sl})$, where $\mathbf{s}_{12}$ is the direction pointing to the origin of the spotlight.

5.4.6 Including Shadows

In the shooting operations, we have ignored the effects of occlusion by surfaces and the effects of absorption and outscatter by the participating medium. For any two points in the environment, we can compute a function $S(\mathbf{x}, \mathbf{x}')$ which returns zero if there is an occlusion by a patch, and one if there are none. The amount of light absorbed and outscattered by the gas is given directly by the transparency $\tau(\mathbf{x}', \mathbf{x})$ between the points (see Equation 5.6). To account for these effects, we multiply each of the form factors by the product of these two terms, i.e.,

$$F_{12} \longrightarrow S(\mathbf{x}_1, \mathbf{x}_2)\tau(\mathbf{x}_1, \mathbf{x}_2)F_{12}.$$

This is actually an approximation if the shooting patch or blob is large, or if the distance between the shooter and the receiver element is small. Some objects which intersect the cone subtended by the disk of the shooting element might be missed by a single ray as shown in Figure 5.14. One solution to this problem is to subdivide the large disk into smaller disks and to shoot from each smaller disk. The transparency can be calculated by using the blob integration technique presented in Section 5.3.1. The occlusion term can be calculated using a standard ray caster.

5.4.7 Multi-Scale Shooting From Blobs

When there are many blobs in the environments, shooting from each blob becomes expensive. The net illumination coming from many blobs is actually very smooth. Therefore, when occlusion is rare, it is possible to shoot from a cluster of blobs rather than from each individual blob. This fact is often used in surface radiosity shooting algorithms where the shooting patch is usually large and the collecting is done on a subdivision of the other patches [14]. We can use our multi-scale blob representation to achieve this (see Section 4.2.2). At each level of the hierarchical tree, we have a set of clusters of blobs. The incoming light caused by the cluster of blobs at the centre of an element "e" can be replaced by the total intensity of the cluster coming from the centre of mass of the blobs, by considering the following approximation:

$$I_{12}^{in} = \sum_{k=1}^{n} J_k(\mathbf{s}_{k,p})F_{k,e} \approx \bar{F}_{k,e} \sum_{k=1}^{n} J_k(\mathbf{s}_{k,e}),$$

where $\bar{F}_{k,e}$ is the average form factor calculated using an average area and the centre of mass of the blobs. The centre of mass of the blobs is calculated as:

$$\bar{\mathbf{x}}_{cm} = \frac{\sum_{k=1}^{n} m_k \mathbf{x}_k}{\sum_{k=1} m_k},$$

the average area $\bar{A}$ is calculated similarly. These quantities can be calculated while building the multi-scale blob representation. With these definitions, the average form factor is given by

$$\bar{F}_{k,e} = \frac{\pi \bar{A}}{\bar{A} + \pi d_{cm,e}^2}\, \mathbf{n}_2 \cdot \mathbf{s}_{cm,e}.$$

To account for shadowing, both the transparency and the occlusion terms can be calculated using only the centre of mass. However, when the blob cluster gets too large, this approximation might be too coarse if there are many occluding patches in the environment. In practice, then, we can use an adaptive algorithm in which more blobs are considered in regions with a variation in shadowing. For example, this can be achieved by comparing the values of the shadow calculations between consecutive levels in the tree.

5.4.8 Extensions

The above algorithm can be extended to intensity fields expanded into an arbitrary number of harmonics. At each shooting operation to a blob, each harmonic must be updated accordingly. However, such an extension becomes rapidly expensive because the number of terms grows quadratically with the order of the approximation. We choose not to implement the general cases since we are able to achieve convincing visual depictions with our reduced expansion. As well, the BRDF of the surfaces can be expanded into spherical harmonics. The generalized algorithm for an arbitrary number of harmonics can then be combined with this formulation. In fact, shooting operations between patches for this type of reflectance functions have been developed by Sillion et. al. [92].

5.5 Multiple Scattering

The optics of [cumulus clouds], though we see them day after day, has not yet been sufficiently investigated. You have to be careful, of course, before accepting the idea that clouds really can absorb light; you should first try to explain everything as if they were solid white objects, and then remember that they really are scattering mists, and, finally consider the possibility of their containing dark dust particles as well. [...] We ought to extend our investigations to other types of clouds as well, and try to explain why rain clouds, for example, are so gray; why in thunder clouds a peculiar leaden colour can be seen side by side with a faded orange. Is it dust? Our knowledge, however, of all these things is so incomplete that we prefer to spur the reader on to begin investigations of his or her own. *M. G. J. Minnaert*

The blob to blob shooting operation is accurate only if the blobs are small or if they are at a large distance from each other. Usually, this situation is not achieved. One possible solution to this problem is to shoot from a subsampling of the blob into smaller blobs. This method, however, becomes prohibitively expensive. Another solution is to attempt to compute the form factor directly, without using the disc approximation. However, this form factor is generally hard to solve, even for simple spherical blobs. To resolve the scattering of light amongst the blobs themselves, we will use a diffusion approximation of the scattering process. This approximation is valid when scattering events abound. These are the exact conditions under which the effects of multiple scattering are important and become visible. We have already encountered this approximation in Section 2.3.1, when the general advection-diffusion equation was derived from the transport equation. The calculation of the multiple scattering can be included into our shooting algorithm by modifying it slightly. We first perform all shooting operations from patches. At this point the intensity due to single scattering events, namely J_{ri}, is resolved (see Equation 5.9). The diffuse source intensity J_d accounts for the multiple scattering within the gas. We compute this term using a diffusion equation. This intensity is then added to the source intensity of each blob and shot to the to the patches of the environment. These steps are iterated until the unshot energy falls below a certain threshold. The algorithm is summarized as follows.

```
set intensity and unshot intensity of blobs and patches to their emission:
    $I = I_{emission}$
    $J = (1 - \Omega)Q$.
shoot from non-physical light sources to all blobs/patches:
        $I = I + \varphi I_{lights}$
        $J = J + \Omega\mathcal{S}\{I_{lights}\}$.
    while unshot energy is below a threshold do
        shoot from each patch to all other patches and all blobs:
            $I = I + \varphi I_{patches}$
            $J = J + \Omega\mathcal{S}\{I_{patches}\}$
        solve for multiple scattering within blobs:
            compute $J_d$
            $J = J + J_d$
        shoot from each blob to all patches:
            $I = I + \varphi J_{blobs}$
    end while
```

Before presenting the diffusion approximation and our implementation, we review similar work done in computer graphics.

5.5.1 Previous Work

To model the effects of multiple scattering, researchers either make simplifying assumptions about the participating medium, or resort to expensive simulations. Rushmeier et al. assume a medium with isotropic scattering properties and derive a radiosity-style algorithm [83]. This method essentially models the interchange of energy between cubical elements (zones) of the environment. Anisotropic effects can also be modelled by discretizing the directions. These methods are known as *Discrete Ordinates* and have been applied to the rendering of participating media by Max and Languénou et al. [55, 43]. Other researchers have used direct Monte-Carlo techniques to simulate the paths of light particles in general environments [3, 70]. None of these models attempt to derive analytical models to account for the effects of multiple scattering. A notable exception is the work of Kajiya and Von Herzen [35]. They model the effects of multiple scattering by expanding the intensity field into a spherical harmonics basis. This method is known as the P_N-method in the transport theory literature, where N is the degree of the highest harmonic in the expansion [16]. Kajiya and Von Herzen derived the general method but used the P_1 expansion in their results as inferred from their statement: "We truncate the so-called 'p-wave', viz. after the $l = 1$ term" [35]. For this particular case, a diffusion-type equation was obtained for the scattered part of the illumination field. Unfortunately, this characterization was obscured by the level of generality of their derivation. Also boundary conditions were not discussed in detail.

5.5.2 The Diffusion Approximation in Radiative Transfer

As was stated in the Section 5.1.3, the diffused intensity is entirely created within the medium, through the phenomenon of scattering. After many scattering events the coefficients of a spherical harmonics expansion of the scattered field tend towards zero as shown in Appendix 5.B. This motivates the main approximation made in the *diffusion approximation*, namely that we can ignore the specular component of the angular dependence of the diffuse source intensity:

$$J_d(\mathbf{x}) = J_d^0(\mathbf{x}) + \mathbf{J}_d^1(\mathbf{x}) \cdot \mathbf{s}.$$

We give another argument justifying this truncation using spherical harmonics in Appendix 5.B. This approximation is usually valid in atmospheric scattering when the ratio of gas volume versus air is higher than 0.01 [31]. As stated in Section 5.1.3 the diffuse source intensity satisfies the transport equation with an additional source term due to the first scatter of the reduced incident intensity:

$$\mathbf{s} \cdot \nabla J_d = -\sigma_t \rho \left(J_d - J_{ri} - (1-\Omega) Q - \Omega \mathcal{S}\{J_d\} \right). \tag{5.20}$$

By substituting the reduced expansion of the diffuse source intensity into this equation we get two equations by grouping terms that have the same order. Indeed the left hand side of Equation 5.20 becomes:

$$\mathbf{s} \cdot \nabla J_d = \mathbf{s} \cdot \nabla J_d^0 + \nabla \cdot \mathbf{J}_d^1.$$

The scattering term on the right hand side can be calculated likewise to be:[4]

$$\Omega \mathcal{S}\{J_d\} = \frac{\Omega}{4\pi} \int_{4\pi} (1 + \bar{\mu}(\mathbf{s} \cdot \mathbf{s}')) \left(J_d^0(\mathbf{x}) + \mathbf{J}_d^1(\mathbf{x}) \cdot \mathbf{s}' \right) \, d\mathbf{s}' = \Omega J_d^0 + \frac{\Omega \bar{\mu}}{3} \mathbf{J}_d^1 \cdot \mathbf{s}.$$

Using these relations and grouping terms that have the same order we get two equations for the coefficients J_d^0 and $\mathbf{J}_d^1$:

$$\nabla \cdot \mathbf{J}_d^1 = -\rho \left(\sigma_a J_d^0 - \sigma_t J_{ri}^0 - \sigma_a Q \right), \tag{5.21}$$

$$\nabla J_d^0 = -\rho \left(\sigma_{tr} \mathbf{J}_d^1 - \sigma_t \mathbf{J}_{ri}^1 \right), \tag{5.22}$$

where J_{ri}^0 and $\mathbf{J}_{ri}^1$ are the first two coefficients of the first scatter due to the reduced incident intensity. The *transport cross section* σ_{tr} is introduced as shorthand notation:

$$\sigma_{tr} = (1 - \Omega \bar{\mu}/3) \sigma_t = \sigma_s (1 - \bar{\mu}/3) + \sigma_a.$$

For constant phase functions, the flux $\mathbf{J}_{ri}$ is equal to zero and the transport cross section is equal to the extinction cross section. These two functions, then, characterize the anisotropy of the diffuse intensity. Equations 5.21 and 5.22 are equivalent to the P_1 equations used by Kajiya and Von Herzen to render their clouds [35]. The diffusion aspect of these equations is, at this point, still hidden. We achieve a single equation for the average diffuse intensity as follows. The average flux can be extracted from the second equation and substituted into the first one to yield a diffusion equation for the average diffuse intensity:

$$\nabla \cdot \left(\kappa \nabla J_d^0 \right) - \alpha J_d^0 + S = 0, \tag{5.23}$$

where we have used the following shorthand notations:

$$\begin{aligned} \kappa(\mathbf{x}) &= \left(\sigma_{tr} \rho(\mathbf{x}) \right)^{-1}, \\ \alpha(\mathbf{x}) &= \sigma_a \rho(\mathbf{x}), \\ S(\mathbf{x}) &= \sigma_t \rho(\mathbf{x}) J_{ri}^0(\mathbf{x}) - \frac{\sigma_t}{\sigma_{tr}} \nabla \cdot \mathbf{J}_{ri}^1(\mathbf{x}) + \sigma_a \rho(\mathbf{x}) Q(\mathbf{x}). \end{aligned}$$

The boundary condition requiring that no diffuse intensity can penetrate the medium at a surface cannot be satisfied exactly, because the diffuse intensity is approximated only by its

[4]We use the following identities: $\int_{4\pi} \mathbf{s}' \, d\mathbf{s}' = 0$, $\int_{4\pi} \mathbf{s} \cdot \mathbf{s}' \, d\mathbf{s}' = 0$ and $\int_{4\pi} (\mathbf{s} \cdot \mathbf{s}') \mathbf{s}' \, d\mathbf{s}' = \frac{4\pi}{3} \mathbf{s}$.

first two moments. Instead, an approximate boundary condition requiring that the total inward flux be zero is appropriate The exact form of this condition is [31]:

$$J_d^0(\mathbf{x}_s) - 2\kappa(\mathbf{x}_s)\frac{\partial}{\partial \mathbf{n}} J_d^0(\mathbf{x}_s) + 2\frac{\sigma_t}{\sigma_{tr}}\mathbf{n} \cdot \mathbf{J}_{ri}^1(\mathbf{x}_s) = 0, \tag{5.24}$$

for all points $\mathbf{x}_s$ lying on the boundary and $\mathbf{n}$ denotes the normal to the surface at point $\mathbf{x}_s$. Once the average diffuse intensity has been calculated, we can compute the average flux from Equation 5.22:

$$\mathbf{J}_d^1(\mathbf{x}) = \kappa(\mathbf{x})\left(-\nabla J_d^0(\mathbf{x}) + \sigma_t \rho(\mathbf{x})\mathbf{J}_{ri}^1(\mathbf{x})\right). \tag{5.25}$$

The flux is thus essentially proportional to the gradient of the average diffuse intensity.

We can now make certain qualitative remarks concerning the phenomenon of multiple scattering from this diffusion equation. The effect of multiple scattering is to smear out the initial source intensity S over time. This initial intensity is equal to self-emission from the medium plus a fraction of the incoming intensity which is neither scattered away nor absorbed by the medium. The diffusion process is stimulated by the diffusion coefficient κ and tempered by the absorption rate α. Therefore, the effects of multiple scattering are most pronounced when the diffusion coefficient is high and the absorption rate is low. Precisely, the diffusion constant is higher for phase functions favouring forward scattering ($\bar{\mu} > 0$) versus backward scattering. The same is achieved when the albedo is close to unity. In particular, clouds have both a high albedo and a strong forward scattering. Multiple scattering is therefore an important phenomenon in clouds.

5.5.3 Numerical Solution of the Diffusion Equation

In Section 2.3.1 we presented two general methods to solve diffusion equations. We implemented both the multi-grid finite difference method and the finite element blob solution. The former method is tractable only in two-dimensions. We implemented these techniques mostly in order to obtain approximate solutions which we can use to assess the accuracy of our blob method. However, these techniques can be of use in particular cases when the medium is contained in a very thin slice.

Multi-Grid Solution in Two-Dimensions

The average diffuse intensity and the source intensity first are sampled on a regular grid of size $M \times M$. These samples are denoted by $I_{i,j}$ and $S_{i,j}$ respectively. The diffusion-absorption operator is then approximated using central differences:

$$(\nabla\kappa\nabla - \alpha)\, I_{i,j} \approx \frac{\kappa_{i+1,j}I_{i+1,j} + \kappa_{i-1,j}I_{i-1,j} + \kappa_{i,j+1}I_{i,j+1} + \kappa_{i,j-1}I_{i,j-1} - 4\kappa_{i,j}I_{i,j}}{4h^2} - \alpha_{i,j}I_{i,j},$$

where $\alpha_{i,j}$ and $\kappa_{i,j}$ are the sampled versions of the absorption and the diffusion constant, respectively and h is the grid spacing. By equating this discretized operator to each sample of the source function, we get a system of equations for the interior points of the domain. After each relaxation step, we update the boundary by discretizing Equation 5.24. Let (i,j) be a point on the boundary, then the variation along the normal is discretized by:

$$\frac{\partial}{\partial \mathbf{n}} I_{i,j} \approx \frac{I_{i',j'} - I_{i,j}}{h},$$

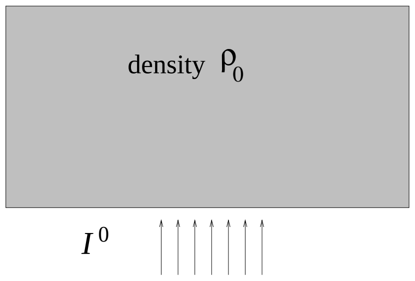

Figure 5.15: Directional light source incident on a constant density slab.

where (i', j') is the closest sample to the boundary along the normal. For example, for the boundary point $(i, 0)$ the closest point is $(i, 1)$. The boundary condition is thus satisfied if the boundary is updated after each relaxation step by:

$$I_{i,j} = \frac{2\kappa_{i',j'} I_{i',j'} + 2h\ \mathbf{n} \cdot \mathbf{Q}_{i',j'}}{h + 2\kappa_{i',j'}},$$

where $\mathbf{Q}_{i,j}$ is the sampled version of the function $\sigma_s/\sigma_{tr}\mathbf{J}_{ri}$. We have implemented the multi-grid scheme outlined in Section 2.3.1.

Blob Finite Element Solution

For three-dimensional problems the multi-grid solution is prohibitively expensive. We can use the numerical blob method of Section 4.3.1 (see Equation 4.11). This equation involves the gradient of the diffusion constant, which in this case is equal to:

$$\nabla\kappa(\mathbf{x}) = \frac{1}{\sigma_{tr}} \nabla \frac{1}{\rho(\mathbf{x})} = -\frac{1}{\sigma_{tr}} \frac{\nabla\rho(\mathbf{s})}{\rho(\mathbf{x})^2}.$$

The finite element equation then becomes:

$$\sum_{k=1}^{N} m_k I^0_{d,k} \left(\rho(\mathbf{x})\nabla^2 G_{2\sigma}(k,j) - \nabla\rho(\mathbf{x}) \cdot \nabla G_{2\sigma}(k,j)(1-\Omega)\sigma_0 G_{2\sigma}(k,j)\right) + \\ m_k S_k \Omega \sigma_0 G_{2\sigma}(k,j) = 0, \qquad j = 1, \cdots, N,$$

where the function $\sigma_0 = \sigma_t \sigma_{tr}$. When N is not too large ($N < 1000$), this system is solved using a direct *LU* decomposition. For larger systems, a relaxation scheme can be used. However, convergence is not usually guaranteed.

Multiple-Scattering in a Constant Density

We present numerical results for the the case of a directional light incident on a slab of constant density. We assume that the direction of propagation is along the positive y-axis $\mathbf{s}_0$ and of magnitude $4\pi I_0(x)$ and that the density is constant and equal to ρ_0 (Figure 5.15). In this specific situation, the reduced incident intensity at each point $\mathbf{x} = (x, y)$ is

$$I_{ri}(\mathbf{x}, \mathbf{s}) = 4\pi I^0(x) \exp(-\sigma_t \rho_0 y)\delta(\mathbf{s} - \mathbf{s}_0).$$

Consequently, the source term due to the reduced incident intensity is equal to:

$$J_{ri}(\mathbf{x}, \mathbf{s}) = \Omega I^0(x) \exp(-\sigma_t \rho_0 y) p(\mathbf{s} \cdot \mathbf{s}_0).$$

The two first coefficients of this function are then

$$\begin{aligned} J^0_{ri}(\mathbf{x}) &= \Omega I^0(x) \exp(-\sigma_t \rho_0 y) \quad \text{and} \\ \mathbf{J}^1_{ri}(\mathbf{x}) &= \Omega I^0(x) \exp(-\sigma_t \rho_0 y) \frac{3}{4\pi} \int_{4\pi} p(\mathbf{s} \cdot \mathbf{s}_0)\, \mathbf{s}\, d\mathbf{s} = \Omega I^0(x) \exp(-\sigma_t \rho_0 y) \bar{\mu}\, \mathbf{s}_0. \end{aligned}$$

The source term appearing in the diffusion equation is then equal to

$$S(\mathbf{x}) = \left(\sigma_t \rho_0 + \frac{\sigma_t \rho_0 \bar{\mu} \sigma_t}{\sigma_{tr}} \right) \Omega I_0(x) \exp(-\sigma_t \rho_0 y) = \left(1 + \frac{\sigma_t \bar{\mu}}{\sigma_{tr}} \right) \sigma_s \rho_0 I^0(x) \exp(-\sigma_t \rho_0 y).$$

The discretized version of the source term can thus be calculated directly once the initial distribution of the intensity $I^0(x)$ is given. Boundary conditions have to be specified on the limits of the computational grid. The conditions for the lateral borders $x = 0$ and $x = h(M+1)$ are given by

$$I_{0,j} = \frac{2\kappa_{1,j}}{h + 2\kappa_{1,j}} I_{1,j} \quad \text{and} \quad I_{M+1,j} = \frac{2\kappa_{M,j}}{h + 2\kappa_{M,j}} I_{M,j},$$

for $j = 1, \cdots, M$. The boundary conditions at the top ($y = h(M+1)$) and the bottom ($y = 0$) of the grid are equal to

$$\begin{aligned} I_{i,0} &= \frac{2\kappa_{i,1} I_{i,1} + 2hQ_{i,1}}{h + 2\kappa_{i,1}} \quad \text{and} \\ I_{i,M+1} &= \frac{2\kappa_{i,1} I_{i,M} - 2hQ_{i,M}}{h + 2\kappa_{i,1}}, \end{aligned}$$

where $i = 1, \cdots, M$. The contribution due to the flux $\mathbf{J}^1_{ri}$ in this particular case amounts to

$$Q_{i,j} = \frac{\sigma_s \bar{\mu}}{\sigma_{tr}} I^0(hi) \exp(-\sigma_t \rho_0 hj).$$

For this particular situation, we have implemented the multi-grid finite difference scheme on a grid of size 512×512 with a "v"-cycle of depth 5. Only three relaxation steps were performed on each level. The solution of the diffusion equation took approximately 30 seconds on an SGI Indigo with an RS4000 processor. Each picture was rendered by assuming that the two-dimensional domain has a certain thickness l. The final intensity for each pixel (hi, hj) is then calculated by:

$$I(hi, hj) = \tau I_{back} + (1 - \tau) \Omega I_{i,j} \quad \text{where} \quad \tau = \exp\left(-\sigma_t \rho_0 l\right).$$

In our pictures we have set $l = 100$ and the background colour I_{back} to blue. The density was set to be equal to 0.1. Figure 5.16 shows the effects of varying the albedo Ω, extinction cross-section σ_t and the first moment $\bar{\mu}$ of the density. Figure 5.17 shows the importance of the boundary conditions. In the picture on the left we have simply set the boundary to be equal to zero. Notice the unnatural decay to zero especially noticeable at the base of the beam.

Comparison with Discrete Ordinates

In order to assess the accuracy of the diffusion approximation, we have compared the results obtained using the multi-grid finite differences method to results obtained using the discrete ordinates method [43]. In Figure 5.18 we compare the results for a constant three-dimensional density defined on a grid of size $32 \times 32 \times 32$. For this situation there is a good agreement between the two methods. In Figure 5.19 the same experiment is conducted on a non-constant density obtained using the spectral synthesis technique of Section 3.3.2. There is a noticeable difference between the two methods at the edge of the cloud. In this region the density is very low and the diffusion approximation does not hold since the amount of scattering is low.

Comparison with Blob Method

In order to evaluate the accuracy of the blob solution we compare it to a solution obtained using the multi-grid scheme. We have computed solutions in two dimensions for two different numbers of blobs (24 and 76). The results are shown in Figure 5.20 and are compared to a multi-grid finite difference solution. The top pictures show the source term for each method. The source term is sampled at the centre of each blob in the finite element method. The pictures at the bottom show the result after diffusion. The results demonstrate that the diffusion approximation does a fairly good job at approximating the solution given by the multi-grid scheme. This is achieved by using a discretization which is an order of magnitude more efficient both in terms of storage (76 versus $512^2 = 262144$ elements) and computation time (0.2 versus 30 seconds). The blob solution could be used in an interactive graphics package.

We have implemented the blob solution finite element method in three-dimensions to calculate the effect of multiple scattering in clouds. Figure 5.21 shows different renderings at different values for the albedo. The phase function was chosen to be constant.

To demonstrate that our rendering algorithm can handle gases with non-constant phase functions, we have computed a little animation of an observer flying around a backlit cloud (see Figure 5.22). The phase function was chosen such that it favours forward scattering as is common in real clouds. More specifically we set the the first moment $\bar{\mu}$ of the phase function to 3/4. We used our stochastic rendering algorithm to add visual detail to the blobs.

Finally, Figure 5.23 shows a still from an animation of a the motion of a single cloud. Notice the self-shadowing due to the sun at the zenith.

Figure 5.16:

Figure 5.17:

Figure 5.18:

Figure 5.19:

Figure 5.20:

Figure 5.21:

Figure 5.22:

Figure 5.23:

5.6 Stochastic Rendering

Chance and randomness did not look like very promising topics for precise investigation, and were in fact shunned by many early scientists. Yet they play now a central role in our understanding of the nature of things. *David Ruelle*

In general, the intensity of light at a given point is a function of the geometry of the scene, the initial lighting conditions (light sources), the reflective properties of surfaces and the scattering properties of a participating medium. If any of these parts are random, then the resulting intensity field will also be random. A stochastic model of the intensity field is a function of a stochastic model for the parts (via the illumination model used). The intensity field can thus be considered as the result of a transformation applied to a random function, as explained in Section 3.2. When this stochastic model can be established, a realization of the random intensity function can be computed. This technique, then, has the potential to speed up the rendering. To our knowledge, no work in computer graphics has been devoted to the calculation of these stochastic models, except for the derivation of reflectance models for surfaces. Following is a brief discussion of these models.

5.6.1 Previous Work

Many reflectance models for surfaces are derived using a stochastic model for the height of the surface. Usually, only the mean value of the intensity reflected from the surface is calculated. Torrance and Sparrow model the surface by a random distribution of normals [96]. Their model has been used in many local illumination models. The fluctuations around the mean are usually accounted for either by adding an arbitrary value through a texture map or by perturbing the mean normal of the surface [4]. Recently, He et al. generalized the work of Torrance and Sparrow to a wave-physics description of light [27]. As far as we know, the only work in computer graphics addressing the problem of calculating the correlation of the reflected intensity field, is the work of Krueger [42]. His calculations become tractable for certain idealized situations (e.g., planar surfaces). The inclusion of the correlation in a reflectance model remains, however, an unexplored area of research in computer graphics. This is mainly due to the mathematical complexity of the task.

In the case of volume rendering, attempts to calculate a stochastic model for the intensity field have not been published. However, Gardner's algorithm to render clouds has the semblance of a stochastic rendering algorithm in that he perturbs the transparency rather than the density of the clouds [20]. Therefore, randomness is added in the rendering instead of in the modelling, as in our algorithm. His model is heuristic and is not based on any well established principle that describes the transfer of light in density fields. Instead, Gardner's model is a modification of a standard illumination model for surfaces. However, because his algorithm is surface based, it has some shortcomings. For example, objects cannot disappear smoothly through his clouds. Recently Kaneda et al. used Gardner's model in conjunction with their atmospheric illumination model [36]. They rendered the most realistic pictures of clouds to date.

In this chapter, we explore only a small set of the possible applications of stochastic rendering. For example, algorithms could be devised to deal with random light sources. We outline the stochastic model used to describe the density field next. This method can be

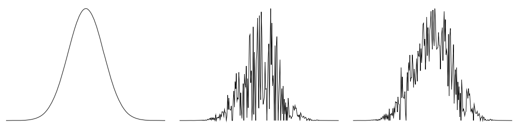

Figure 5.24: $\sigma^2_{\rho,k} = 0$, $\sigma^2_{\rho,k} = \epsilon\bar{\rho}_k$ and $\sigma^2_{\rho,k} = \epsilon F_\eta(\bar{\rho}_k)$

seen as an alternative to the blob warping described in the previous sections to add visual detail.

5.6.2 Stochastic Model for the Density Field

We interpret the blob representation for the density field as its mean value at all points in the volume. The mean is therefore a superposition of the mean values $\bar{\rho}_k$ of each individual blob:

$$\bar{\rho}(\mathbf{x}) = \sum_{k=1}^{N} m_k W(\mathbf{x} - \mathbf{x}_k, \sigma) = \sum_{k=1}^{N} \bar{\rho}_k(\mathbf{x}),$$

In other words, the random function can be seen as a weighted sum of "random blobs". This interpretation is very convenient for our purposes. Consequently, we describe the correlation functions of each blob, instead of the correlation function of the entire density field. We want the variance of each blob to be proportional to its mean. A natural choice, then, is to make the variance directly proportional to the mean: $\sigma^2_{\rho,k} = \epsilon\bar{\rho}_k$. However, with this choice, variations tend to become too large near the centre of the blob, which is unrealistic. We expect the variance to drop off with the mean but not to grow to exceedingly large values. We can avoid the latter by "clamping" the values of the mean: $\sigma^2_{\rho,k}(\mathbf{x}) = \epsilon F_\eta(\bar{\rho}_k(\mathbf{x}))$. The clamping function is defined as:

$$F_\eta(t) = \begin{cases} t & \text{if } t < \eta \\ \eta & \text{otherwise.} \end{cases}$$

Figure 5.24 shows the effect of the clamping function on the resulting random blob. The cutoff parameter η and the magnitude ϵ are either provided by a user or are estimated from data. We assume that the random blobs are transformations of a single homogeneous random function with zero mean Q (see Section 3.2.2): $\rho_k(\mathbf{x}) = \bar{\rho}_k(\mathbf{x}) + \sigma_{\rho,k}(\mathbf{x})Q(\mathbf{x})$. The correlation function of each blob therefore has the following form:

$$C_{\rho,k}(\mathbf{x}'; \mathbf{x}'') = C_Q(\mathbf{x}'' - \mathbf{x}')\sigma_{\rho,k}(\mathbf{x}')\sigma_{\rho,k}(\mathbf{x}''). \tag{5.26}$$

We assume that the correlation function C_Q of the homogeneous random function Q is a Gaussian with standard deviation α and zero mean. This assumption is important and is used later when we derive a stochastic model for the intensity field.

5.6.3 Derivation of the Stochastic Rendering Model

We have seen that the amount of light coming from a participating medium and reaching an observer is given by a linear combination of the background intensity and the average source function (Equation 5.11). Because the density field is random, the transparency and the average source function appearing in the equation are also random.

Transparency

To derive a stochastic model of the transparency, we make the assumption that all rays originate from the xy plane and that their directions are in the positive z direction: $\mathbf{x}_u = (x_0, y_0, u)$. In other words, we assume an orthographic projection. The integral in the equation for the transparency (Equation 5.6) then becomes a function of the location (x_0, y_0) in the xy plane:

$$\begin{aligned} \Gamma(x_0, y_0) &= \int_0^b \rho(x_0, y_0, u)\, du = \sum_{k=1}^N \int_0^b \rho_k(x_0, y_0, u)\, du \\ &= \sum_{k=1}^N \Gamma_k(x_0, y_0). \end{aligned}$$

Each integral Γ_i is a random field which has a well defined mean and correlation (see Equation 3.16):

$$\begin{aligned} \bar{\Gamma}_k(x_0, y_0) &= \int_0^b \bar{\rho}_k(x_0, y_0, u)\, du, \\ C_{\Gamma,k}(x_0, y_0; x_1, y_1) &= \int_0^b \int_0^{b'} C_{\rho,k}(x_0, y_0, u; x_1, y_1, u') du du'. \end{aligned} \tag{5.27}$$

In other words, the mean and correlation of Γ_k are directly related to the mean and correlation of the density ρ_k through an integral. Using the fact that the correlation of the homogeneous random function Q is Gaussian, the correlation of the integral Γ_k becomes (see Equation 5.26):[5]

$$\begin{aligned} C_{\Gamma,k}(x_0, y_0; x_1, y_1) &= C_Q(x_1 - x_0, y_1 - y_0, 0) \times \\ & \int_0^b \int_0^{b'} C_Q(0, 0, u' - u)\sigma_{\rho,k}(x_0, y_0, u)\sigma_{\rho,k}(x_1, y_1, u')\, du du' \\ &\approx C_Q(x_1 - x_0, y_1 - y_0, 0) \int_0^b \sigma_{\rho,k}(x_0, y_0, u)\sigma_{\rho,k}(x_1, y_1, u) du. \end{aligned} \tag{5.28}$$

The approximation appearing in this equation follows from the fact that the support of the correlation function C_Q is much smaller than the support of the variance $\sigma^2_{\rho,k}$:

$$\int_0^{b'} C_Q(0, 0, u' - u)\sigma_{\rho,k}(x_1, y_1, u')\, du' \approx \sigma_{\rho,k}(x_1, y_1, u).$$

In the limiting case where the correlation is a delta function, the above relation is exact. Equation 5.28 demonstrates that the spatial structure of each integral Γ_k is essentially the same as the spatial structure of a "slice" of the density field. They differ in that they have a different mean and variance. Specifically, the variance is given by:

$$\sigma^2_{\Gamma,k}(x_0, y_0) = C_Q(0, 0, 0)\epsilon \int_0^b F_\eta(\bar{\rho}_k(x_0, y_0, u))\, du. \tag{5.29}$$

If both the mean and the correlation are computable, a realization $\Gamma_{k,\omega}$ for each of the integrals can be generated:

$$\Gamma_{k,\omega}(x_0, y_0) = \bar{\Gamma}_k(x_0, y_0) + \sigma_{\Gamma,k}(x_0, y_0) Q_\omega(x_0, y_0, 0), \tag{5.30}$$

[5] We use the separability of a Gaussian: $C_Q(x, y, z) = C_Q(x, 0, 0)C_Q(0, y, 0)C_Q(0, 0, z)$.

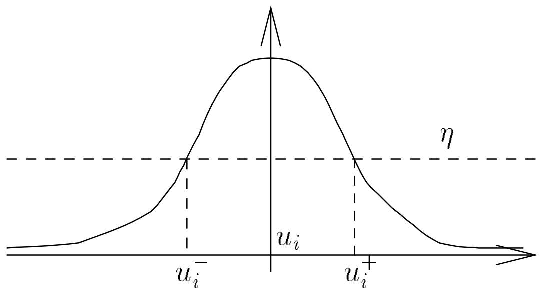

Figure 5.25: Clamping of the blob.

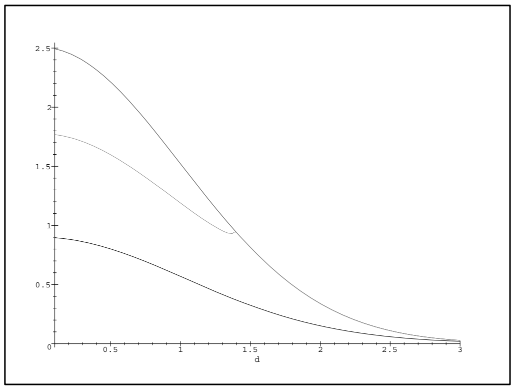

Figure 5.26: Variance $\sigma^2_{\Gamma,k}$ versus the distance d_k for different values of the clamping η.

where Q_ω is a realization of the homogeneous random function. A realization for the transparency then is equal to $\exp(-\kappa_t \sum_{k=1}^{N} \Gamma_{k,\omega}(x_0, y_0))$. The mean value can be calculated as in Section 5.3.1. The integral of the variance can be calculated similarly and is described next.

Let u_k^- and u_k^+ be the two parameter values on the ray between which the mean density of the blob is clamped, i.e., for which $m_k W(|\mathbf{x}_u - \mathbf{x}_k|, \sigma) = \eta$ (see Figure 5.25). These values may not be defined if the maximum of the mean is smaller than the clamping value, in which case we set $u_k^- = u_k^+ = c_k$. The integral can be computed by integrating over each interval separately:

$$\int_0^b F_\eta(\bar{\rho}_k(\mathbf{x}_u))\, du = \int_0^{u_k^-} \bar{\rho}_k(\mathbf{x}_u)\, du + \int_{u_k^-}^{u_k^+} \eta\, du + \int_{u_k^+}^{b} \bar{\rho}_k(\mathbf{x}_u)\, du.$$

Each of these integrals can be computed using the blob integration algorithm of Section 5.3.1. For the case of a Gaussian smoothing kernel, the clamping values are given by $u_k^{\pm} = c_k \pm \Delta u_k$, where:

$$\Delta u_k = \sqrt{2 \log \left(\frac{m_i}{(2\pi)^{\frac{3}{2}}\, h^3 \eta} \exp\left(\frac{-d_i^2}{2\sigma^2} \right) \right)}.$$

Realizations of the integrals Γ_k can therefore be generated efficiently using Equation 5.30: $\Gamma_{k,\omega} = \bar{\Gamma}_k + \sigma_{\Gamma,k} Q_\omega(\mathbf{x}_{c_k})$. The choice of $\mathbf{x}_{c_k}$ (the point closest to the centre of the blob) guarantees that the perturbation is sampled on a plane for each frame and that the samples are chosen coherently from frame to frame. The latter is important in animated sequences, where the viewing conditions change. Artifacts can, however, appear for example when rotating around a cloud since the texture will not vary uniformly across the cloud. Indeed the texture at the center of the cloud will not change and the change is greatest at the edge of the cloud. In our animations, however, we have noticed no such artifacts. In Figure 5.26 we show plots of the variance $\sigma^2_{\Gamma,k}$ versus the distance to the centre of the blob d_k for different values of the clamping value η. As expected, the variance increases as the distance decreases, i.e., when the ray is near the centre of the blob. The plain line in the graph indicates no clamping and is equal to the mean $\bar{\Gamma}_k$. It is interesting to note that Gardner varied the threshold of his transparency function in a similar way. To derive the statistical description of the integrals Γ_k, we assume that the viewing rays are parallel. This is of course not the case in most practical situations. However, if the size of each blob is small relative to the image, then the rays emanating from a point are approximately parallel across each blob.

Average Source Intensity

The statistics of the average source function are much more complicated to calculate. In this work we will assume that the average source function is only a function of the average of the density field. There are sound justifications for this assumption. First, the average source function is in fact a convolution of the source function by the "weighting" function $\tau(0,u)\rho(\mathbf{x}_u)\kappa_t$ along the ray. Second, the source function itself involves an integral accounting for scattering from different directions. This scattering acts to smooth the intensity field. In fact we modelled part of it as a diffusion process. The same smoothing phenomenon caused by indirect illumination is also observed in radiosity environments [10].

5.6.4 Algorithm

Following, we give an overview of our stochastic rendering algorithm. It is a modification of the algorithms given in Sections 5.3.1 and 5.3.2.

```
gamma = 0
for each blob intersecting the ray do
    Calculate av_gamma = $\bar{\Gamma}_k$ and s_gamma = $\sigma_{\Gamma,k}$
    Sample $Q$ at midpoint: Q = $Q_\omega(\mathbf{x}_{c_k})$
    Update integral: gamma = gamma + av_gamma + s_gamma*Q
end for
Calculate transparency: tau = exp(-$\kappa_t$*gamma)
Calculate av_J = $\bar{J}(\mathbf{x}_0, \mathbf{s})$ from the blob representation
using the algorithm described in Section 5.4
Combine: I = tau*$I_{out}(\mathbf{x}_b, \mathbf{s})$ + (1-tau)*av_J
```

Whenever a shadow ray is computed, only the transparency tau must be calculated. The complexity of this algorithm is $O(N)$, where N is the number of blobs. Unlike volume rendering algorithms, our algorithm is independent of the resolution of random perturbation of the density field. In the next section, we describe results obtained using our algorithm.

5.6.5 Results

If we had always lived in a land where there are no clouds, only low mist or haze, what would we have thought if a sranger had come to tell us that in his country these layers of mist rise high in the sky and become purple, crimson, scarlet and gold. *John Ruskin*

We implemented the stochastic rendering algorithm into a standard ray-tracer. Prior to rendering, we precompute the tables for the transparency and we generate a realization of the homogeneous random field Q. The latter is generated using the inverse spectral method of Section 3.3.2. The spectral density is obtained by taking the inverse Fourier transform of the correlation function. As stated in Section 3.3.2, this method has the advantage that the resulting realization can be defined and is periodic on a regular three-dimensional lattice. Therefore it can be evaluated efficiently using trilinear interpolation and is defined for all points in space. The artifacts caused by the periodicity of the field are not visible in our images because the field is used only to perturb the illumination. We used a table of size $32 \times 32 \times 32$ in all of our results. A user has control over the shape of the density field by specifying the position, centre and mass of each blob of the mean density (see Section 5.6.2). The perturbation is controlled by providing the magnitude ϵ of the variance, the clamping value η and the correlation function of the homogeneous field Q. In all the results we have assumed a constant phase function p, i.e., scattering is constant in all directions. Next we specify some of our results obtained using the stochastic rendering algorithm.

We model clouds in a way similar to Gardner. An individual cloud is generated by randomly placing blobs in an ellipsoid provided by a user. The size and mass of each blob are made inversely proportional to their distance from the centre of the ellipsoid to ensure that the density is maximum in the centre of the cloud. Clusters of individual clouds can be generated by randomly generating such ellipsoids. In Figure 5.27 we show four pictures of the same cloud with different values for the magnitude ϵ and the clamping value η. Clouds in the left column have a lower perturbation ϵ than the ones on the right; and the ones on the bottom have a lower threshold value η than the ones on the top. Figure 5.28 shows one of the clouds with self shadowing, two different clusterings of such clouds and an areal view of downtown on a foggy day. Clouds are characterized by a high amount of scattering, therefore we have set the albedo close to one. Figure 5.29 shows four frames from an animated sequence of a cloud interacting with Toronto's CN Tower. This demonstrates that our model handles the interaction of clouds with solid objects. This is an improvement over Gardner's illumination model.

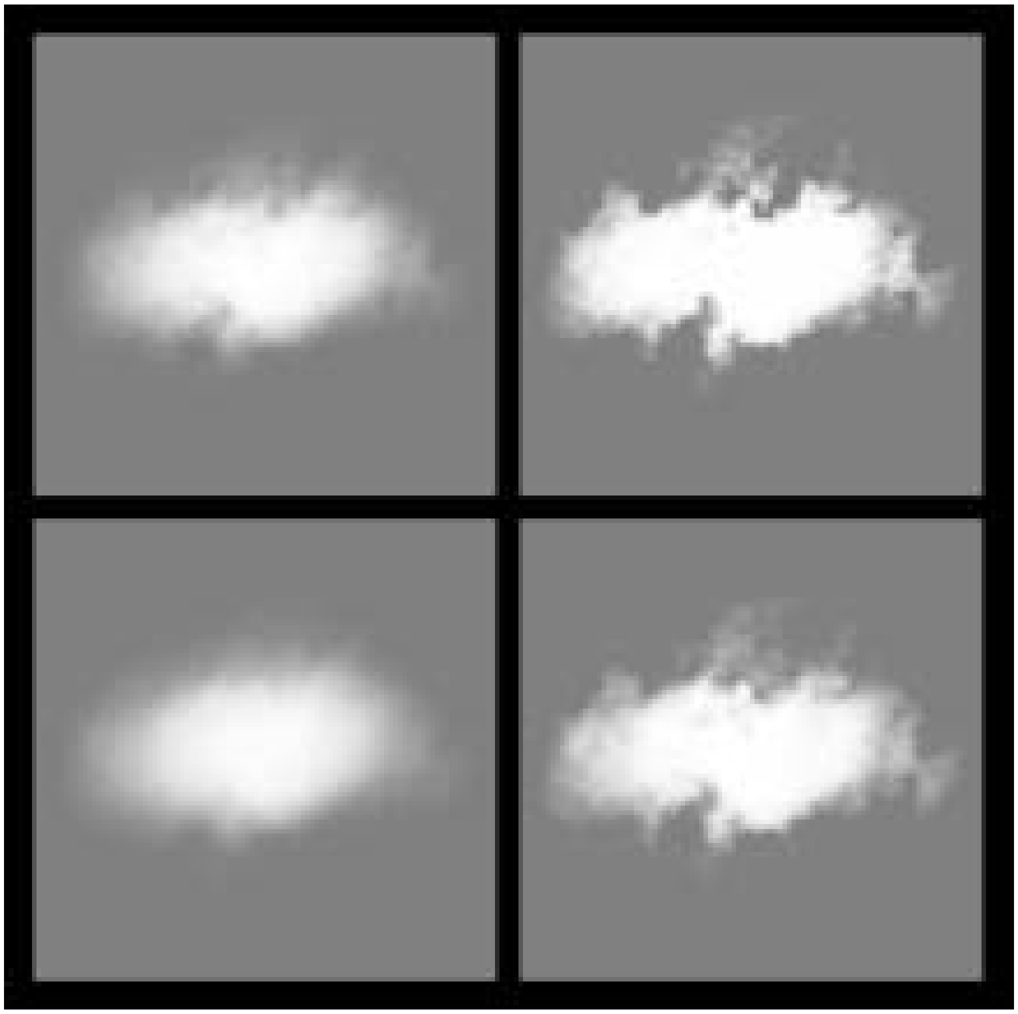

Figure 5.27: Different values for the perturbation ϵ and the threshold η. From top to bottom and left to right: (a) small ϵ high η, (b) high ϵ high η, (c) small ϵ small η, (d) high ϵ and small η.

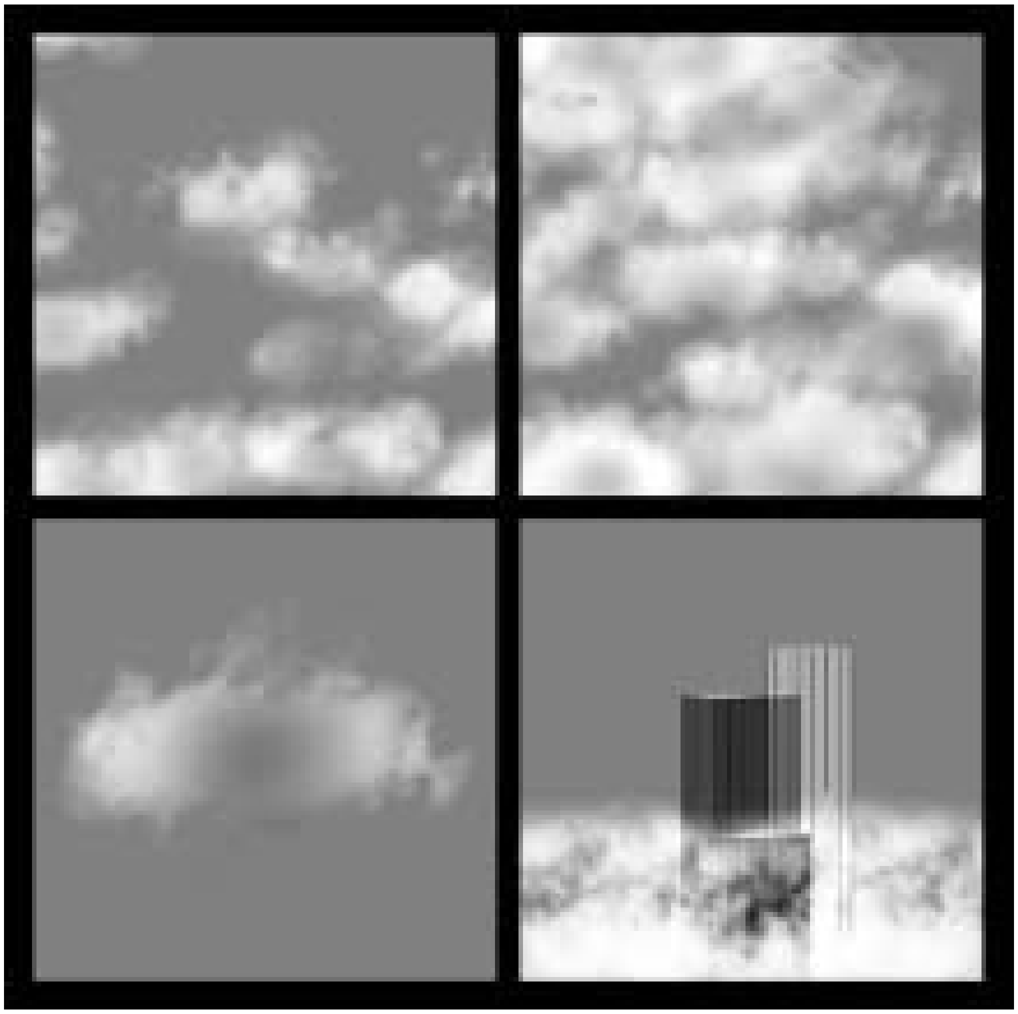

Figure 5.28: Different pictures of clouds created using the stochastic rendering algorithm.

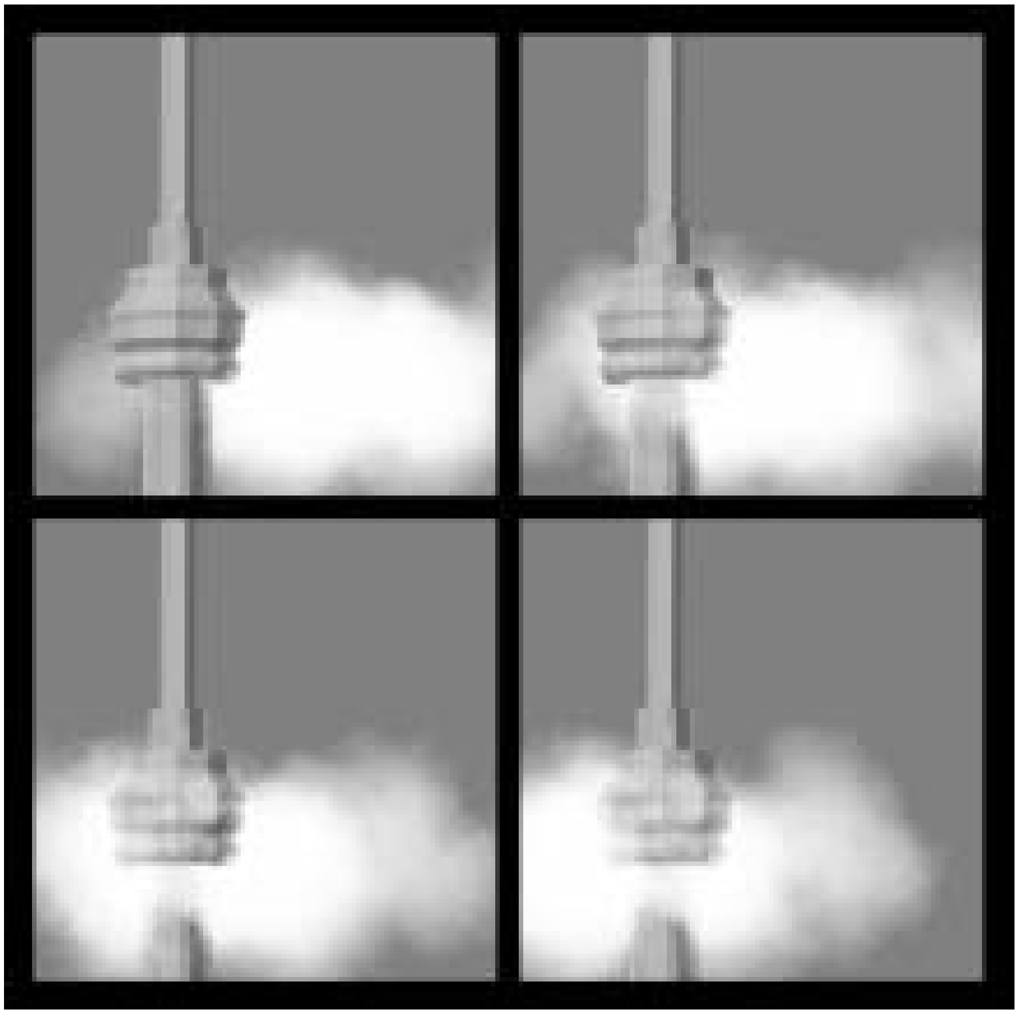

Figure 5.29: Four frames of an animation of a cloud interacting with the CN tower. Notice the smooth disappearance of the tower in the cloud.

5.7 Rendering Hair

The appearance of hair shares many characteristics of density fields. Indeed, light is scattered and absorbed by each hair strand. A noticeable difference with gases, however, is that the scattering depends strongly on the orientation of each hair. Consequently, the phase function does not only depend on the angle between the incoming and outgoing angles. We generalize our rendering algorithm to handle *fuzzy segments* instead of blobs. Each hair is then approximated by a set of adjacent fuzzy segments. Prior to presenting our model we review previous work done in this area.

5.7.1 Previous Work

Researchers have modelled hair either by modelling each hair explicitly or by modelling the hairs as a solid texture. The latter approach is useful to model short fur-like hair. Perlin and Hoffert directly volume rendered the solid texture to depict fur [73]. Kajiya and Kay on the other hand modelled the fur as a distribution of microfacets and derived an illumination model from this distribution [34]. Although, their results are impressive the method is too expensive for practical applications. Explicit models of hair fall into two-categories. Either the hair is modelled as an opaque cylinder or as a translucent segment. The explicit cylinder model was used by Anjyo et al. to render opaque thick hair [1]. The model works well in depicting dark hair but fails to render the translucency exhibited by blonde hair for example. The latter problem has been solved using depth buffers by both LeBlanc [45] and Rosenblum et al. [82]. In both algorithms shadowing is achieved by projecting the hairs into a z-buffer centred at each light source. Various shadowing effects are achieved by combining the various depth buffers. This method has the advantage that the hardware rendering capabilities of graphics workstation can be used to calculate the shadows. However, it is prone to aliasing when the number of hairs is very large. The aliasing can be avoided by either perturbing the samples or by filtering adjacent values, at the cost of loss of detail and increase in computation time. Our approach is similar to theirs, as we pre-filter each hair segment in world space instead of screen space. Our method, however, uses the efficient blob integration techniques of Section 5.3 and the shooting algorithm of Section 5.4 to compute the shadowing and the illumination of the hairs respectively.

5.7.2 Hair Illumination Model

Our illumination model is very similar to the ones used in most previous models. Hair is visible to us because each hair strand both reflects and refracts light. The effects of refraction and multiple reflections can be modelled as the scattering of light within a density field of hairs. This model is, however, not complete, since highly directional reflections from hair are often visible. This effect is particularly apparent for greased hair. We assume that the width of each hair strand is thin and that the distribution of reflected light due to the incoming intensity coming from a direction $\mathbf{s}$ is equal to the "reflected cone" aligned with the direction $\mathbf{t}$ of the hair as illustrated in Figure 5.30. This is equivalent to adding a term p_{spec} to the phase function which depends on the direction of the hair strand:

$$p_{spec}(\mathbf{s}\cdot\mathbf{s}',\mathbf{t}) = p^{spec}\delta(\mathbf{s}'\cdot\mathbf{t} - (-\mathbf{s}\cdot\mathbf{t})) = p^{spec}\delta((\mathbf{s}'+\mathbf{s})\cdot\mathbf{t}). \tag{5.31}$$

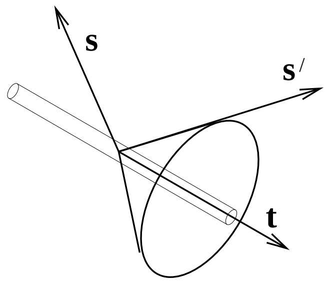

Figure 5.30: The intensity reflected from a hair is approximated be a cone.

Consequently, the shooting operations from patches and light sources to the hair blobs have to be changed accordingly. We represent each hair as a density field defined around a segment $\mathbf{z}_v = \mathbf{z}_0 + v\mathbf{t}$:

$$H(\mathbf{x}) = W(\mathrm{dist}_H(\mathbf{x}), \sigma),$$

where dist_H gives the shortest distance to the hair segment and W is a smoothing kernel (see Section 4.1). The point closest on the segment (assuming the direction $\mathbf{t}$ is normalized) is equal to the projection of the vector $\mathbf{x} - \mathbf{z}_0$ onto the segment:

$$\mathbf{z}_{closest} = \mathbf{z}_0 + ((\mathbf{x} - \mathbf{z}_0) \cdot \mathbf{t})\mathbf{t}.$$

In the case when this point does not lie within the hair segment, $\mathbf{z}_{closest}$ is one of the endpoints of the segment. Consequently, the distance is equal to:

$$d_H(\mathbf{x}) = |(\mathbf{x} - \mathbf{z}_0) + ((\mathbf{x} - \mathbf{z}_0) \cdot \mathbf{t})\mathbf{t}|.$$

Following, we give an integration method to compute the transparency due to an ensemble of such fuzzy segments.

5.7.3 Fuzzy Segments Integration

The results presented in this subsection are a generalization of the spherical blob integration to fuzzy segments. In order to compute the optical depth due to a single fuzzy segment along a ray $\mathbf{x}_u = \mathbf{x}_0 + u\mathbf{s}$, the following integral has to be calculated

$$\Gamma(a, b) = \int_a^b W(d_H(\mathbf{x}_u), \sigma)\, du.$$

Similarly to the blob integration technique we approximate this integral by (see Equation 5.18):

$$\Gamma(a, b) \approx \frac{b-a}{R} \frac{d_3}{R} W(d_1, \sigma),$$

where d_1 is the shortest distance of the ray to the segment, d_3 is the half-distance that the ray traverses through the truncated fuzzy segment (see Figure 5.31) and u^* is the closest point on the ray to the segment. The shortest distance to the segment is calculated by the condition that the difference between the ray and the segment $\mathbf{x}_u - \mathbf{z}_v$ has to be orthogonal to both the direction of the ray and the direction of the segment:

$$(\mathbf{x}_u - \mathbf{z}_v) \cdot \mathbf{s} = 0 \quad \text{and} \quad (\mathbf{x}_u - \mathbf{z}_v) \cdot \mathbf{t} = 0.$$

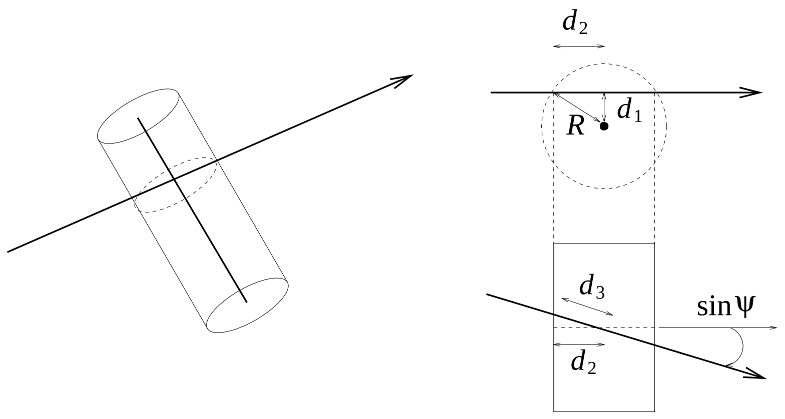

Figure 5.31: Geometry of the integration of a fuzzy segment along a ray.

Solving for u and v one obtains:

$$\begin{aligned} u^* &= (\mathbf{x}_0 - \mathbf{z}_0) \cdot (\mathbf{s} - (\mathbf{s} \cdot \mathbf{t})\mathbf{t})/(1 - (\mathbf{s} \cdot \mathbf{t})^2), \\ v^* &= (\mathbf{x}_0 - \mathbf{z}_0) \cdot \mathbf{t} + u^*(\mathbf{s} \cdot \mathbf{t}). \end{aligned}$$

The denominator $1 - (\mathbf{s} \cdot \mathbf{t})^2$ vanishes when the ray and the segment are aligned and in this case the shortest distance is given by $d_1 = |\mathbf{x}_0 - \mathbf{z}_{closest}|$ for example. In general, the closest distance is then given by $d_1 = |\mathbf{x}_{u^*} - \mathbf{z}_{v^*}|$. From the geometry of Figure 5.31 it follows that the distance along the ray is given by

$$d_3 = d_2/\sin\psi = \sqrt{R^2 - d_1^2}/\sin\psi,$$

where R is the radius of the truncated blob and ψ is the angle between the direction of the segment and the direction of the ray, and hence

$$\sin\psi = \sqrt{1 - (\mathbf{s} \cdot \mathbf{t})^2}.$$

The optical depth due to many segments is obtained by summing up the contributions of each of them.

5.7.4 Hair Rendering

The hair can be rendered by adding shooting operations from the patches and the light sources towards the extremities of each hair segment. The intensity on each point of the segment is then interpolated using the parameter v^* on the segment. Each shooting operation is identical with an additional specular component due to the reflected cone (see Equation 5.31). Indeed, from a patch, the contribution to the source intensity due to the specular scattering is given by

$$J_{spec} = I_1^{out} \Omega F_{12} p_{spec}(\mathbf{s} \cdot \mathbf{s}_{12}, \mathbf{t}).$$

Similarly for the light sources. The integration of the source intensity is then similar to the one for the blobs given in Section 5.3.2.

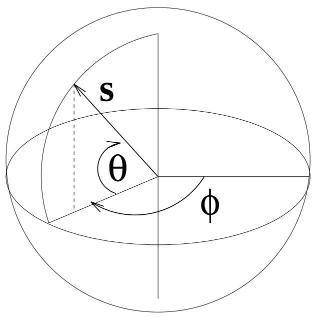

Figure 5.32: Spherical coordinates used in the definition of the spherical harmonics basis functions.

Appendix 5.A Spherical Harmonics

Spherical harmonics form an orthonormal basis for the space of functions defined on the unit sphere. Each direction $\mathbf{s}$ can be represented by two angles $\theta \in [-\pi/2, \pi/2]$ and $\phi \in [0, 2\pi]$ as illustrated in Figure 5.32. The spherical harmonics are then defined from the *Legendre polynomials* $P_l(\mu)$ as:

$$Y_{l,m}(\mathbf{s}) = Y_{l,m}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta)e^{im\phi},$$

where the functions P_l^m are the *associated Legendre polynomials* of degree l and order $m \geq 0$ defined by:

$$P_l^m(\mu) = (-1)^m(1-\mu^2)^{m/2}\frac{d^m}{d\mu^m}P_l(\mu).$$

For negative m, the spherical harmonics are defined through the following relation:

$$Y_{l,m}(\mathbf{s}) = (-1)^{-m}Y^*_{l,-m}(\mathbf{s}).$$

For each positive order l, we have $2l+1$ harmonics with $m = -l, \cdots, l$. The Legendre polynomials are defined by the following recurrence relations:

$$(l+1)P_{l+1}(\mu) = (2l+1)\mu P_l(\mu) - lP_{l-1}(\mu), \quad l \geq 2,$$

with the first two polynomials equal to

$$P_0(\mu) = 1 \quad \text{and} \quad P_1(\mu) = \mu.$$

The Legendre polynomials are themselves an orthogonal basis for functions on the interval $[-1,1]$:

$$\int_{-1}^{1} P_l(\mu)P_{l'}(\mu)\,d\mu = \frac{2}{2l+1}\delta_{ll'}.$$

The Legendre polynomials are related directly to the spherical harmonics through *the addition theorem*:

$$P_l(\mathbf{s}\cdot\mathbf{s}') = \sum_{m=-l}^{l}\left(\frac{4\pi}{2l+1}\right)Y^*_{l,m}(\mathbf{s}')Y_{l,m}(\mathbf{s}). \tag{5.32}$$

Appendix 5.B Proof of Angular Smoothing Due to Multiple Scattering

Both the phase function and the angular component of the intensity field can be expanded into spherical harmonics:

$$\begin{aligned} p(\mathbf{s}\cdot\mathbf{s}') &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} p^l Y^*_{l,m}(\mathbf{s}')Y_{l,m}(\mathbf{s}), \\ I(\mathbf{x},\mathbf{s}) &= \sum_{l'=0}^{\infty}\sum_{m'=-l'}^{l'} I^{l',m'}(\mathbf{x})Y_{l',m'}(\mathbf{s}), \end{aligned}$$

specifically $p^0 = 4\pi$ and $p^1 = 2\pi\bar{\mu}/3$. Because of the orthonormality property of the spherical harmonics, each single scatter event becomes a simple multiplication by the coefficients of the phase function:

$$\mathcal{S}\{I\}(\mathbf{x},\mathbf{s}) = \sum_{l',m'}\sum_{l,m} p^l I^{l',m'}(\mathbf{x})Y_{l,m}(\mathbf{s})\frac{1}{4\pi}\int_{4\pi} Y^*_{l,m}(\mathbf{s}')Y_{l',m'}(\mathbf{s}')\,d\mathbf{s}' = \sum_{l,m}\frac{p^l}{4\pi}I^{l,m}(\mathbf{x})Y_{l,m}(\mathbf{s}).$$

The accumulative effect on the intensity field of n scattering events at a location $\mathbf{x}$ can be expressed through the scattering functional $\mathcal{S}$ (see Eq. 5.3) as

$$\mathcal{S}^n\{I\}(\mathbf{x},\mathbf{s}) = \mathcal{S}^{n-1}\{\mathcal{S}\{I\}\}(\mathbf{x},\mathbf{s}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\left(\frac{p^l}{4\pi}\right)^n I^{l,m}(\mathbf{x})Y_{l,m}(\mathbf{s}).$$

This last expression tends towards $I^{0,0}(\mathbf{x})$ as n tends towards infinity. This is a consequence of the fact that each coefficient p^l is strictly smaller than 4π in absolute value, with the exception of the first one. This follows directly from the definition of each coefficient:

$$\begin{aligned} |p^l| &= \left|2\pi\int_{-1}^{+1} p(x)|P_l(x)|\,dx\right| \le 2\pi\int_{-1}^{+1} p(x)|P_l(x)|\,dx \\ &< 2\pi\int_{-1}^{+1} p(x)\,dx = 4\pi. \end{aligned}$$

the last inequality follows from the fact that $|P_l(x)| < 1$ for $l > 0$. This demonstrates that the dependence of the intensity field diminishes as the number of scatter events n increases.

Appendix 5.C Efficient Integration of the Transport Equation

The algorithm of Section 5.3.2 can be improved by making the following additional approximations: (see Equation 5.19)

$$J(\mathbf{x}_u,\mathbf{s}) \approx = \frac{\sum_k m_k J_k(\mathbf{s})W(d_k,\sigma)}{\sum_k m_k W(d_k,\sigma)},$$

on each interval. Here, the values of the smoothing kernel can be computed before calculating the integral. At each interval, either a new blob overlaps or an overlap ends. The source

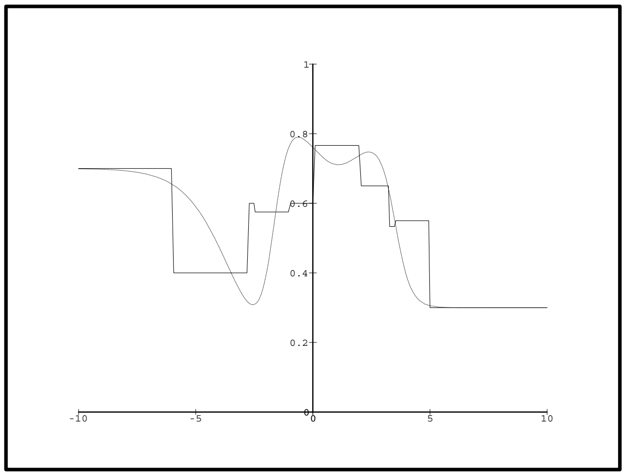

Figure 5.33: Linear Approximation of the source intensity when the approximation of Appendix 5.C is used.

intensity can thus be updated for each consecutive interval by either adding the intensity of the blob which starts to overlap the interval,or subtracting the intensity to end the overlap. The approximation is shown in Figure 5.33. It is fairly coarse, however the visual results are comparable with those obtained from the algorithm of Section 5.3.2 at a computation time which is an order of magnitude faster. The algorithm follows.

```
I_d = 0
tau_tot = 0
gamma = J = rho = 0
for $j = 0, \ldots, K_i - 1$ do
    $k$ = index of blob that enters or exits the interval
    if blob enters then
        m = $m_k W(d_k, \sigma)$
        gamma = gamma + m*$\frac{1}{\Delta_k}$
        J = J + m*$J_k$(s)
        rho = rho + m
    else if blob exits then
        m = $m_k W(d_k, \sigma)$
        gamma = gamma - m*$\frac{1}{\Delta_k}$
        J = J - m*$J_k$(s)
        rho = rho - m
    end if
    tau = exp(-$\sigma_t(u_{j+1} - u_j)$*gamma)
    I_d += tau_tot*(1-tau)*J/rho
    tau_tot = tau_tot*tau
    if tau_tot < EPS then exit loop
end for
I = tau_tot*$I_{out}(\mathrm{x}_b, \mathrm{s})$ + I_d
```

Care has to be taken with regard to numerical instabilities. When subtracting the values from the variable rho, gamma and J the previous values should cancel. In general, with finite precision, the result is non zero. When many blobs intersect the ray, small residuals can accumulate to produce visible artifacts. In practice then, we clamp the variables to zero whenever they fall below a given threshold. We effectively remove the inner loop. The algorithm therefore runs in time $O(|L|)$ when there is no warping. The warping of the blob is included in the calculation of the value of the smoothing kernel, as in Section 5.3.2.

Chapter 6

Depiction of Gases, Fire and Hair

Modelling is certainly not a scientific endeavor, even though it is customary to report it as such. Successful modelling is nearly always reported as a more or less logical reconstruction of occurrences. Derivations are presented in logical sequences which has little relationship to the manner in which the modelling effort progressed. Generally many attempts were made to produce results and only the one finally chosen was reported. The reason why this is so is that the modeller wants to make his work acceptable to the scientific community. Also, it provides a convenient, acceptable frame for reporting. As a result, an unrealistic view is presented as to what modelling actually is and how it is done. *J.J. Leendertse*

In this chapter we outline our visual simulations, based on the methodology and techniques presented in the previous chapters. Specifically, models are developed to depict turbulent gases, the spread and evolution of fire and the motion and appearance of hair. The evolution of each of these phenomena can be modelled as a advection-diffusion process subjected to a motion field. As well, the appearance of these phenomena can be calculated using the density field rendering algorithms discussed in the previous chapter. In the first section, we review previous work. Then, in the second section, our general methodology is described. The subsequent sections illustrate concrete applications of this methodology related to the depiction of gases, fire and hair.

6.1 Previous Work

We review relevant work dealing with the depictions of both the motion and the rendering of gases, fire and hair. Previous work in this area falls approximately into two categories. In the first group, researchers animate the density field directly using three-dimensional texture maps. In the second approach, the density field is animated through the use of motion (wind) fields. Our work falls into the latter category.

Solid textures were first introduced by Perlin and Peachy [72, 71]. These textures were employed to add visual detail to smooth surfaces either by modifying the intensity, or by perturbing the normal [4]. Solid textures were later utilized to perturb a transparent layer around objects and were called *hypertexture* [73]. These hypertextures were applied in depicting fire balls and hair, among other effects. Kajiya and Kay also modelled hair, by deriving an illumination model from a distribution of micro-facets defined within a layer around an object [34]. In particular, they created a very realistic picture of a teddy bear. The animation of the parameters of a solid texture was first explored by Ebert [17]. In the same spirit as Perlin's work, Ebert proposed many procedural models which were discovered empirically. Sakas used statistical models of turbulence to derive meaningful parameters [85]. These parameters are based on energy cascade models similar to the ones considered in this thesis (see Section 3.4.2). However, we apply these models to the simulation of motion fields, not the density field. In both Ebert's and Sakas work the texture is rendered using voxel traversal algorithms. This method is disadvantageous, especially in Ebert's model, since real-time rendering is not yet possible on standard workstations.

In the second approach, the motion of the density field is subjected to a motion field. One of the first models to use this approach is Reeve's particle system model [78]. He used his particle system to generate the explosions in the opening scene of "Wrath of Khan" [68]. He also applied his model to the generation of grass and leaves [79]. Although he did not state it explicitly, his grass model could have been applied to the modelling of hair. Sims extended this particle systems work by using particles which were blurred both in time and space [93]. This rendering approach is therefore similar to our blob method [93]. These models, however, did not attempt to derive a realistic rendering algorithm for the depiction of these particle systems. Also, the motion fields used in these simulations are deterministic and the depiction of turbulent motion is rather difficult. Shinya and Fournier were the first to introduce stochastic wind fields into animations [89]. Their fields were two-dimensional and they invoked the Taylor Hypothesis (see Section 3.4.2) to model the evolution over time. They used this model to animate various phenomena such as grass. However, they did not apply the model to the animation of gaseous phenomena. Stam and Fiume generalized the turbulent wind field synthesis technique to three-dimensional random vector fields evolving over time and proposed an efficient rendering algorithm based on blobs of density. On the other hand, Chiba et al. modelled two-dimensional turbulent fields as the superposition of point vortex fields evolving over time [11]. They applied their model to the animation of gases and fire. They also developed a model for the spread of fire by subdividing the environment into cells. The propagation of the fire is determined by both the amount fuel and the amount heat at each cell. Their model is restricted to two-dimensional domains. Perry and Picard used a spread model from the fire engineering literature to model the spread directly onto each surface [74]. The spread is entirely determined by its direction and the angle of the surface.

The animation of hair has only received minimal attention in the graphics community. Both Anjyo and Rosenblum et al. model each hair strand as a flexible beam [1, 82]. The simulations that they present are convincing for thick hair exhibiting inertia. However, for light hair their method is computationally expensive. In addition, inter hair collisions have to be calculated on a pair-wise basis and are therefore expensive. Real-time simulations can only be achieved with a small number of hairs.

6.2 Interlocking Models

Fluids such as gases, water or flames are described by their density, velocity and temperature fields. For arbitrary fluids, the evolution of these quantities is described by the hydrodynamic equations presented in Section 2.3.2. The quantities describing the fluid are strongly coupled through these equations. For example, the temperature field introduces gradients in the velocity field, but the velocity field advects and diffuses the temperature field. In the case of reacting fluids such as fire, additional equations are needed to account for the underlying chemical reactions. The full physical simulation of this set of equations is extremely expensive. Their nonlinear nature and the wide range of scales required would severely strain computational speed and memory available on even the most powerful computers. As outlined in the introduction of this thesis, even if such a simulation were available it would be of limited interest to computer graphics, because the user's ability to control the phenomenon would be missing. The degree of user control must be balanced against the need for turbulent behaviour of a gas, which is difficult to model without an underlying physical or statistical model driving it. In our simulations, we allow the user to control the phenomenon through the specification of a motion field. We assume that this motion field is incompressible and consequently its density can be considered to be constant. Once the motion field is specified, the evolution of both the temperature of the fluid and the density of a substance immersed in the wind field can be approximated by an advection-diffusion process. At each instant of time, the phenomenon can be rendered from its density and temperature using one of the algorithms of the previous chapter. The visual simulation of the phenomenon is thus broken down into three components: a model for the motion field, an advection-diffusion process and a rendering model. The separation of the simulation into these components has several advantages over direct physical simulations. For each component we can trade accuracy for speed. Typically, an animator first starts with an approximate method of solution. Once a particular motion has been sketched, the animation can be fine tuned by using more accurate techniques, to achieve higher realism for example. The three models are now described in detail.

6.3 Motion Fields

A motion field is a vector field $\mathbf{u}(\mathbf{x}, t)$ which varies over space and time. We separate this motion field into two components: a smooth field $\mathbf{u}_1$ and a turbulent field $\mathbf{u}_2$. The smooth field corresponds to the global motion of the phenomenon and physically corresponds to the large scales. An animator utilizes the smooth field to sketch the global motion. The turbulent field corresponds to the small scale detail, which adds irregularity and hence realism to the motion. The small scale motion is usually hard for an animator to specify directly. For this reason, we model the turbulent field as a random function whose macroscopic parameters are specified by the animator. The resulting field is obtained from a composition of these two fields. The most straightforward composition is a simple addition of these two terms:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_1(\mathbf{x}, t) + \mathbf{u}_2(\mathbf{x}, t). \tag{6.1}$$

In this situation, the resulting field does not satisfy the hydrodynamic Equations 2.8, 2.9 and 2.10, even if both components do. Indeed, if we assume that both fields satisfy the hydrodynamic equations, then a simple calculation shows that the conditions

$$(\mathbf{u}_1 \cdot \nabla)\mathbf{u}_2 = 0 \quad \text{and} \quad (\mathbf{u}_2 \cdot \nabla)\mathbf{u}_1 = 0,$$

are sufficient for the addition of the two fields to satisfy the hydrodynamic equations. The first term represents the effect of the large scales on the turbulent ones, and the second term represents the effect of the small scales on the smooth field. These terms are in general non-zero. For example, the turbulence behind an object in movement is strongly dependent on

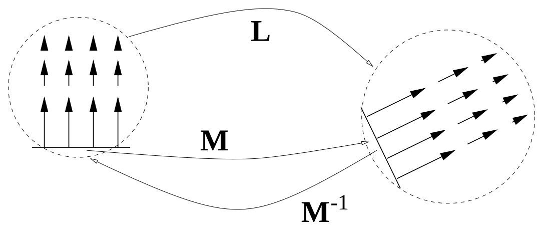

Figure 6.1: Affine transformation of a field.

the global motion of the object. The influence of the turbulent scales on the smooth scales is usually more difficult to visualize. But, it is well known that tiny disturbances at the small scales can modify the larger scales. In our model, however, this interaction is undesirable since the smooth scales are specified entirely by the animator. Hence, we assume that there is no effect from the smaller scales on the smooth field:

$$(\mathbf{u}_2 \cdot \nabla)\mathbf{u}_1 = 0.$$

Since the turbulent scales are random and the smooth field is entirely specified by the animator, we can attempt to derive models for the effect of the smooth fields on the turbulent scales. The general form of our motion field is then:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_1(\mathbf{x}, t) + \mathcal{F}\{\mathbf{u}_1, \mathbf{u}_2\}(\mathbf{x}, t), \tag{6.2}$$

where $\mathcal{F}$ is some mapping. Ensuing, we describe the smooth component in more detail.

6.3.1 Smooth Motion Fields

We model the smooth motion field as a superposition of simple primitive fields. This model is similar to the wind field models which were used to animate particle systems [93] and to animate leaves and other objects [102]. As in geometric modelling, the primitive fields are defined in a local coordinate system. Then, particular instances are obtained from affine transformations of the primitive field. To evaluate the field we have to transform back to the local coordinate system and evaluate the primitive field. The vector so evaluated has to be transformed back through the linear part $\mathbf{L}$ of the affine transformation $\mathbf{M}$ (see Figure 6.1). Precisely, an instance $\mathbf{u}_i$ is obtained from a primitive field $\mathbf{u}_p$ via:

$$\mathbf{u}_i(\mathbf{x}, t) = \mathbf{L}\mathbf{u}_p(\mathbf{M}^{-1}\mathbf{x}, t),$$

and the linear part of the affine transformation is the upper left 3×3 block of matrix $\mathbf{M}$. Our primitive smooth fields are all centred at the origin and usually have a directional component which is pointed in the positive z-direction. In order to localize the fields in space, we add a decaying factor depending on a "distance" d from a "source" region, defined accordingly for each primitive. The decay will be modelled by a function $W(d, \sigma)$ similar to the smoothing kernel considered in Chapter 4. In the remainder of this subsection $\mathbf{x} = (x, y, z)$. Next we describe the different primitive fields that we consider in this thesis. The first three fields are depicted in Figure 6.2.

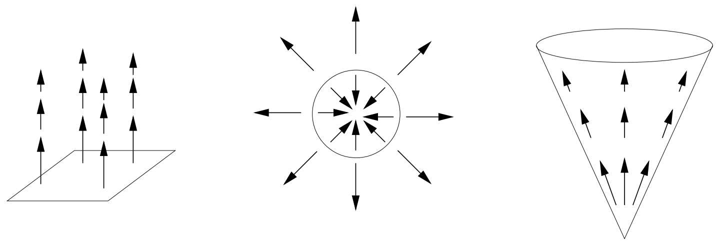

Figure 6.2: Three examples of smooth fields

Directional Fields

Many phenomena, such as steam rising from a coffee cup, are characterized by a strong directional component. Steam rises because of the heat field created by the hot coffee. So, we consider a simple directional motion field which decays with the distance from a given surface. In local coordinates, the direction is upward in the z-direction and zero outside the semi-infinite column defined by the unit square on the xy-plane.

$$\mathbf{u}_{dir}(x,y,z) = \begin{cases} W(z,\sigma)(0,0,1)^T & \text{if} \quad (x,y) \in [-1,1]\times[-1,+1] \text{ and } z>0, \\ 0 & \text{elsewhere.} \end{cases}$$

The sudden cutoff can be smoothed out by multiplying the field by a smooth function which is zero outside of the unit square. In practice, however, the sudden cutoff has not produced serious visible artifacts.

Radial Fields

Radial fields are used to define repulsive fields around objects. The field is directed outwards from the unit sphere and drops off with the distance $d = |\mathbf{x}| - 1$ from the surface of the sphere. Inside the sphere the field is directed inwards. The direction of the forces can be reflected by having a negative magnitude. The definition is:

$$\mathbf{u}_{rad}(\mathbf{x}) = \begin{cases} \frac{d}{|d|} W(d,\sigma) \frac{\mathbf{x}}{d+1} & \text{if} \quad d \neq 0 \\ 0 & \text{elsewhere.} \end{cases}$$

For non-spherical objects, the affine transformation is used to fit a small set of ellipsoidal fields around an object. This problem is akin to the calculation of bounding ellipsoids around arbitrary objects to speed up ray casting algorithms [8].

Cone Fields

Heat fields are characterized by an outward expansion of the directional field, due to the diffusive nature of the propagation of temperature. We must then define a field which is zero outside of a cone and has a directional dropoff inside. The exact definition is given by:

$$\mathbf{u}_{cone}(x,y,z) = \begin{cases} W(|\mathbf{x}|,\sigma) \frac{\mathbf{x}}{|\mathbf{x}|} & \text{if} \quad x^2+y^2<z^2. \\ 0 & \text{elsewhere.} \end{cases}$$

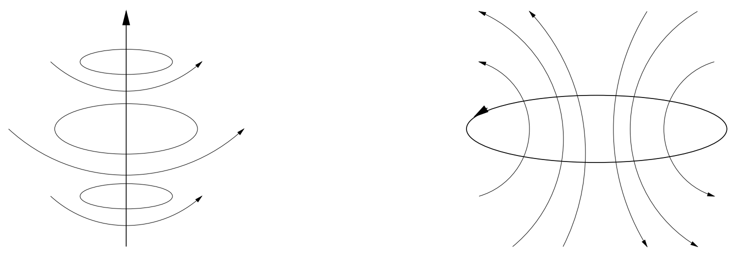
Figure 6.3: Two examples of simple vortex fields

Vortex Fields

An eddie of a velocity field is usually depicted as a small whorl around a given direction and is called a *vortex*. In particular we can consider the velocity field obtained from a set of such vorticies. Indeed, animators have used such combinations of vortex fields to evoke turbulent fields [93]. In this thesis, we present two vortex fields which an animator can use to add particular eddies to the turbulent motion field. These two fields are depicted in Figure 6.3. Vortex fields can be used also for large scale effects such as the evolution of a tornado or the generation of mushroom clouds in an explosion. The first vortex field is the velocity resulting from a single vorticity defined on a line. In local coordinates, the velocity turns around the z-axis counter-clockwise. The definition of the field is given by:

$$\mathbf{u}_{lvort}(\mathbf{x}) = \frac{1}{2\pi}\frac{W(|\mathbf{x}|,\sigma)}{x^2+y^2}(-y,x,0).$$

The second vortex field is obtained by considering the field which is obtained by turning around a circle of radius R. In local coordinates the circle lies in the xy-plane and the field is equal to:

$$\mathbf{u}_{cvort}(\mathbf{x}) = \frac{1}{4\pi}\frac{(zx, zy, (x_1-x)x+(y_1-y)y)}{\sqrt{x^2+y^2}((x-x_1)^2+(y-y_1)^2+z^2)^{3/2}},$$

where

$$(x_1,y_1,z_1) = \frac{R}{\sqrt{x^2+y^2}}(x,y,0)$$

is the point closest to $\mathbf{x}$ on the circle. The derivation of these equations is given in Appendix 6.A of this chapter.

6.3.2 Turbulent Motion Fields

As stated in the introduction of this chapter, we model the turbulent component of the motion field by a random function. Algorithms to synthesize realizations of incompressible and isotropic random functions were given in Section 3.4. Specifically, we choose the spectral synthesis technique since it generates a periodic motion field defined on a four-dimensional lattice. The periodicity is an important feature of our turbulent field because it permits the definition of the field for any point in space and time. Also, the field can be computed efficiently using 4-linear interpolation from nearby points on the grid. A simple algorithm which performs this interpolation is given in Appendix 6.B of this chapter. In practice we have

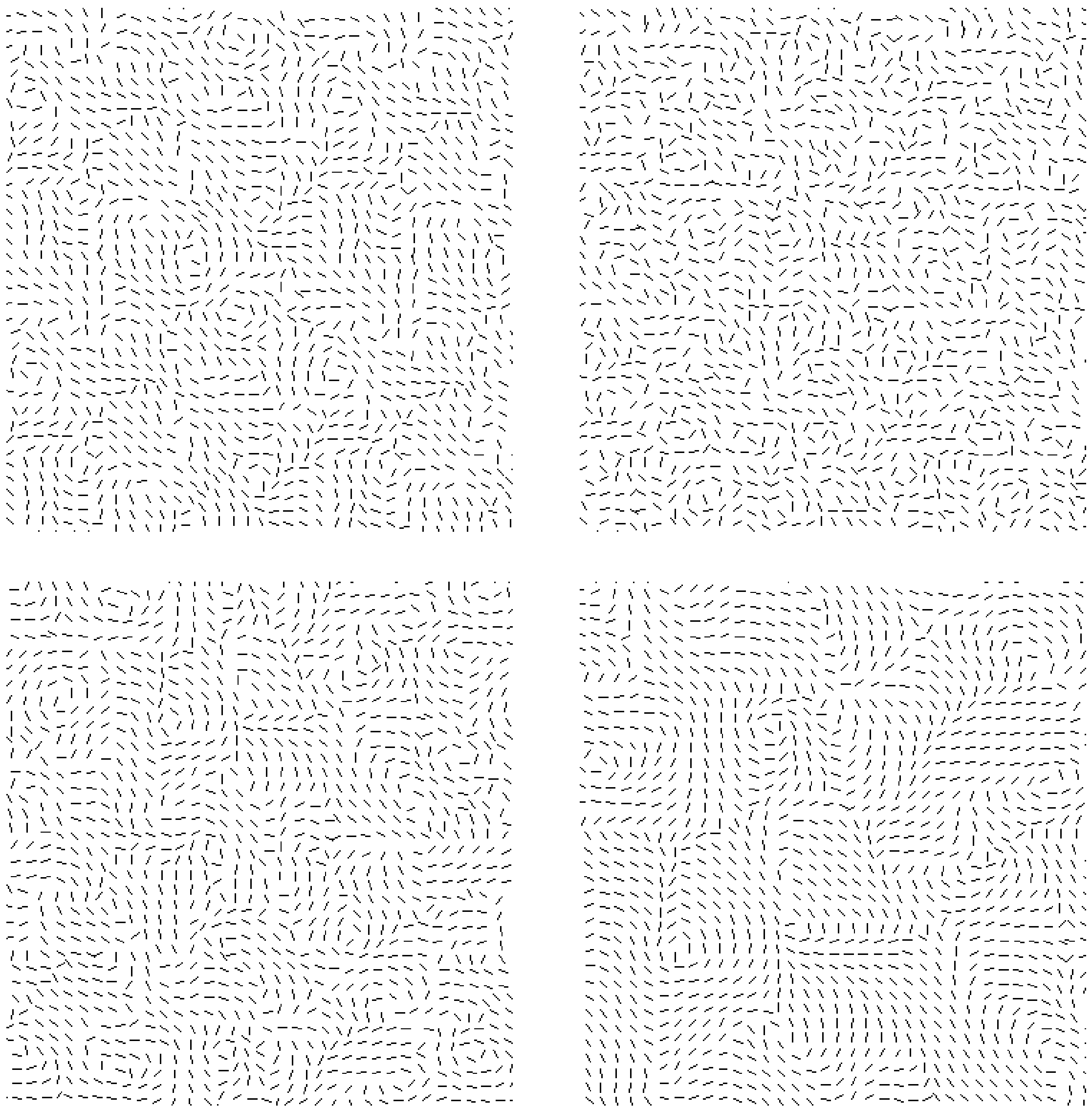
Figure 6.4: Turbulent fields with different spatial structures. From top to bottom and left to right: (a) Large L, (b) Smaller L, (c) Same L as (b) but larger grid size, (d) Same as (c) but even larger grid size.

opted for a resolution of 16 in both space and time. This resolution is satisfactory for most of our simulations. No substantial improvement in the quality of the motion was observed by increasing the resolution. Also, no periodic artifacts were visible at this resolution.

Control Parameters

The macroscopic features of the turbulence are entirely determined by the shape of the energy spectrum (Equation 3.27). For the modified Kolmogorov spectrum these parameters are the size of the largest eddie L and the standard deviation σ of the temporal spread function G_σ (see Equation 3.28). The size of the largest eddie determines the spatial structure of the turbulence. Large values of L correspond to fields with larger spatial structures, i.e., eddies. The converse is true for smaller values of this parameter. The effect of changing this parameter is equivalent to the "roughness" parameter used in the generation of fractal terrain [99]. In the top of Figure 6.4 we show two fields with different "roughness" parameters. The

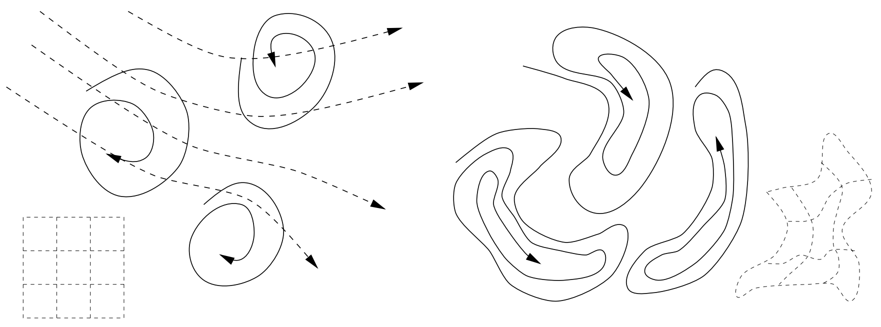
Figure 6.5: The smooth field has the effect of warping the grid of the turbulent field.

temporal spread σ is the equivalent of the roughness parameter for the variation of the field over time. Large spread values correspond to a highly chaotic motion field over time as many temporal frequencies contribute to the total energy. Conversely for small spreads, the field is smoother over time. In practice, the same effects can be achieved by changing the spacing of the turbulence grid once the field has been computed. The two fields on the bottom of Figure 6.4 were actually obtained by increasing the grid size of the top-right field. In particular, the grid spacing of the turbulence can be changed in real time and consequently an animator can explore different settings and results in real time. This speeds up the design process considerably. In exact terms, more than one turbulence grid can be used in the same environment, creating a nested hierarchy of grids.

Interaction with the Smooth Fields

Usually, the simple superposition (Equation 6.1) of the smooth field with the turbulent one has worked well in practice. However, we briefly discuss different possible compositions. In some cases, the turbulence is a function of the magnitude of one of the smooth fields. For example, the turbulence caused by heat becomes stronger and of a different nature the closer the proximity. This can be achieved by masking the turbulence by the smooth field (see Equation 6.2):

$$\mathcal{F}\{\mathbf{u}_1, \mathbf{u}_2\}(\mathbf{x}, t) = f\left(|\mathbf{u}_1(\mathbf{x}, t)|\right) \mathbf{u}_2(\mathbf{x}, t).$$

where f can be any arbitrary function provided by an animator. For example, a simple decaying exponential can be used to smoothly dissipate the scales of the turbulence. In many cases, however, the smooth fields do not only change the magnitude of the turbulence but also change its spatial and temporal behaviour. Usually, the structure of a turbulent field subjected to a forcing term is not isotropic. In fact, the turbulent field is distorted by the smooth fields. This is similar to the problem that we encountered with our regular spherical blobs. To account for the distortion, we can use a similar solution by backwarping the domain along thee smooth vector fields (see Figure 6.5). In fact, the non-zero advecting term $(\mathbf{u}_1 \cdot \nabla)\mathbf{u}_2$ appearing in the hydrodynamic equations describes these deformations. As in the blob warping method, we can evaluate the field by backtracing through the smooth field:

$$\mathcal{F}\{\mathbf{u}_1, \mathbf{u}_2\}(\mathbf{x}, t) = \mathbf{u}_1\left(\mathbf{x} - \int_0^t \mathbf{u}_2(\mathbf{x}(s), s)\, ds, t\right).$$

Consider the case when the smooth field $\mathbf{u}_1$ is a constant direction. Then the turbulent field is given by:

$$\mathbf{u}_2(\mathbf{x} - \mathbf{u}_1 t, t),$$

i.e., the turbulence is translated along the direction. This effect is evident when watching smoke or clouds moving along a given wind direction. In general, when the smooth field varies over space, the field must be recalculated at each time step. Max et al. have developed algorithms to advect grids in unsteady velocity fields and have obtained good results for short time ranges [56].

6.4 Advection-Diffusion Processes

We model the evolution of a phenomenon subjected to a motion field using an advection-diffusion process. The equation which describes this class of processes was derived in Chapter 2 (Equation 2.5). Although this description is approximate, it is well suited to the visual depiction of phenomena as varied as steam and human hair. The resulting simulations are not necessarily physically accurate, but capture the essential characteristics of the phenomenon. Following, we present these equations for the evolution of the density and the temperature field of a gaseous phenomena.

6.4.1 Density and Temperature

A gaseous phenomenon is entirely described by its density, velocity and temperature fields. We suppose that the velocity of the gaseous phenomenon is entirely determined by the motion fields described in the previous section. Also we have assumed that the motion field $\mathbf{u}(\mathbf{x}, t)$ is incompressible and its density $\rho_f(\mathbf{x}, t)$ is therefore presumed to be constant[1]. This density should not be confused with the density ρ of gas particles, which is not constant and evolves over time. For example, in the case of clouds, the density of water droplets is non-constant and fluctuates wildly. The evolution of the density distribution over time under the influence of this motion field is given by the following advection diffusion equation:

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho = \kappa_\rho \nabla^2 \rho - \alpha_\rho \rho + S_\rho.$$

In the general formulation of this equation given in Section 2.3.1, κ_ρ characterizes the diffusion of the particles as they interact with the molecules of the motion field. In practice, however, this term models the diffusion caused by the small scales of the motion field which are not explicitly modelled by the turbulent field, i.e., caused by the subgrid scales. The source term and the absorption rate both depend on the phenomenon under consideration and are described for different cases in the next subsection.

An accurate description of the evolution of the temperature field is usually very complicated. We use a simplified model which is often used in theoretical investigations of fire propagation [9, 13]. These descriptions assume that the kinetic energy is negligible compared to the energy released by the heated gas, and that pressure and buoyancy fluctuations are

[1]Although some of our smooth fields are not strictly incompressible, they are approximately so since their speed is an order of magnitude slower than the speed of sound [47].

small. In this setting, the evolution of the temperature field is described by the following advection-diffusion equation:

$$\rho_f C_p \left(\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) = \kappa_T \nabla^2 T - L_T + S_T,$$

where C_p is the specific heat at a constant temperature of the gas and κ_T is the thermal diffusivity. Although the thermal diffusivity usually depends on the temperature, we assume that it is constant. The loss of temperature L_T is dominated by radiation which is characterized by the absorption cross section of the gas. The specific heat C_p of the gas describes how effectively its temperature can be lifted. The source term depends on the heat released during the creation of the gas and is discussed in the upcoming subsection. The density of the motion field ρ_f is assumed to be constant as stated in the beginning of this section. The source term depends on a simple chemical reaction described next.

6.4.2 The Creation and Spread of Fire

In the case of fire, the source term can be related to a simple combustion equation. In fact, fires result from the combustion of fuels and oxidizers. As the molecules of these compounds come into contact at a sufficiently high temperature, a chemical reaction becomes possible. The burning compounds resulting from this reaction are called the *flame*. We are not interested in a complete physical model of this reaction, but rather are interested in those mechanisms that are essential to a good visual representation of it. Given the density ρ_{fuel} of the fuel and its temperature T_f at a given point, the source term S is given by the *Arrhenius formula*. Assuming a constant concentration of oxidants [9],

$$S_{\rho,flame} = \nu_a \exp\left(-\frac{T_a}{T_{fuel}} \right) \rho_{fuel}. \tag{6.3}$$

T_a is the activation temperature, which is directly related to the energy E_a released during the reaction by $T_a = E_a/R$, where R is the universal gas constant. The term ν_a is a "frequency", depending on the exact nature of the combustibles, characterizing the rate of the reaction.

The initial temperature of the flame is related to the heat released during the reaction modelled by Equation 6.3. The source term appearing in the diffusion equation for the evolution of the temperature of the flame is then equal to

$$S_{T,flame} = C_p S_{\rho,flame} T_a.$$

Fire loses heat mainly through radiation. This loss is equal to the divergence of the radiant heat flux:

$$L_{T,flame} = \int_0^\infty \int_{4\pi} \mathbf{s} \cdot \nabla I_\lambda(\mathbf{x}, \mathbf{s})\, d\mathbf{s}\, d\lambda = \int_0^\infty \int_{4\pi} \sigma_{t,\lambda} \rho(\mathbf{x}) J(\mathbf{x}, \mathbf{s})\, d\mathbf{s}\, d\lambda,$$

where the last identity follows from the scattering equation (Equation 5.4). With flames, we can assume that most of the heat is radiated through emission and we consequently set the albedo to zero. In this case the source intensity J is equal to the black body emission

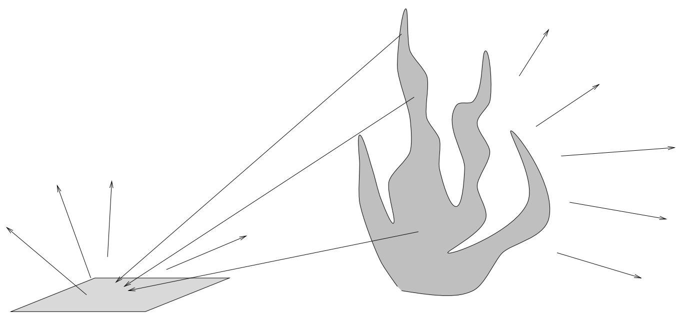
Figure 6.6: The flame and the fuel of the solid exchange heat mainly through radiated heat.

multiplied by the emission spectrum E_λ (see Equation 5.1). The loss of temperature is then given by:

$$L_{T,flame}T = -\int_0^\infty \int_{4\pi} \rho\sigma_{t,\lambda}E_\lambda B_\lambda(T)d\mathbf{s}\; d\lambda = -4\pi\rho\bar{\sigma}_a\sigma T^4,$$

where *mean absorption cross-section* $\bar{\sigma}_a$ is defined by

$$\bar{\sigma}_a = \frac{\int_0^\infty \sigma_{t,\lambda}E_\lambda B_\lambda(T)\; d\lambda}{\int_0^\infty B_\lambda(T)\; d\lambda} = \frac{1}{\sigma T^4}\int_0^\infty \sigma_{t,\lambda}E_\lambda B_\lambda(T)\; d\lambda.$$

Therefore, this constant depends on the emissive and the absorptive properties of the flame. The loss of temperature with time, then, is faster than the linear loss term usually found in advection-diffusion equations. Indeed

$$\frac{\partial T}{\partial t} = -CT^4 \quad \text{implies} \quad T(t) = \frac{1}{(3Ct + T_0^{-3})^{1/3}},$$

where T_0 is the initial temperature of the flame. However, we can still use the blob simulation technique by using this function instead of the exponential one.

Under our assumptions, the creation of the flame is entirely determined by the density and the temperature of the fuel. The spread of fire is then a function of the values of these fields. The evolution of a gaseous fuel, such as natural gas, is also described by an advection-diffusion equation. Solid fuels such as logs, paper or a match are modelled by defining a "thin layer" of fuel around the surface defining the solid. In particular, the shape of the object cannot change as the object burns. However, most objects do change when they burn and eventually collapse into ashes. These effects are, however, difficult to model, so we concentrate on a simple model first. Both the density and the temperature of the fuel are then defined on a subdivision of the surface into patches. The evolution of these fields is governed by a diffusion equation, since the object is supposed to be unaffected by the wind field.

The loss of the density of the fuel is proportional to the source term of the flame. Indeed, in the combustion reaction, part of the fuel is transformed into the flame, hence

$$L_{\rho,fuel} = -S_{\rho,flame}.$$

The source term $S_{\rho,fuel}$ of the fuel is specified by an animator. For example, a flame thrower can be modelled by having a directional source term at the tip of the gun. The source term for the temperature of the fuel is determined from the heat of nearby radiating flames as illustrated in Figure 6.6. The source term for the temperature of the fuel is then the fraction of this radiation which is incident upon the fuel. This is similar to a shooting operation from a blob to a patch (solid fuel) or to a blob (liquid fuel) presented in Sections 5.4.3 and 5.4.4 respectively where a form factor F_{12} is calculated. By adding up the contributions from all wavelength, the total heat due to a nearby blob of a radiating flame is then given by (for a solid fuel)

$$S_{T,fuel} = \rho_{fuel}\bar{\sigma}_a F_{12}\sigma T^4_{flame},$$

where $\bar{\sigma}_a$ is the average absorption cross-section of the fuel. In practice, we have used the multi-scale blob representation of the flame, and shoot only from a small number of blobs. The fuel also loses heat due to radiation similarly to the flame. In this case the loss term appearing in the diffusion equation for the evolution of the temperature of a gaseous fuel is

$$L_{T,fuel} = 4\pi\rho_{fuel}\bar{\sigma}_a\sigma T^4.$$

For a solid fuel the factor 4π is replaced by the factor 2π.

6.4.3 Smoke Creation

When the flame cools off, soot particles cluster together to form heavy smoke particles. Also at the time of the reaction, other compounds other than the flame are created which propagate further into the environment. To the best of our knowledge, there are no analytical models to describe this phenomenon in the literature. Therefore, we have modelled the creation of smoke by using a simple Arrhenius type equation. Once the temperature of the flame drops off, the density of smoke particles increases:

$$S_{\rho,smoke} = \nu_b \exp\left(-\frac{T_{flame}}{T_b}\right)\rho_{flame},$$

where T_b is a typical temperature at which smoke creation becomes important. The frequency ν_b models the rate of smoke creation.

6.4.4 Method of Solution

To solve the advection-diffusion equations, we use the blob representation and the blob simulation algorithm given in Section 4.3.2. Both the blob and temperature fields for each gas in the environment are expanded into a blob representation:

$$\begin{aligned} \rho(\mathbf{x},t) &= \sum_{k=1}^{N} m_k(t)W(\mathbf{x}-\mathbf{x}_k(t),\sigma) \\ T(\mathbf{x},t) &= \frac{1}{\rho(\mathbf{x},t)}\sum_{k=1}^{N} m_k(t)T_k(t)W(\mathbf{x}-\mathbf{x}_k(t),\sigma). \end{aligned}$$

To update both these fields, the algorithm of Section 4.3.2 can be applied directly. In environments containing fire, the evolution of the flames, the fuel and the smoke have to be coupled together in order to yield a simulation. The algorithm is:

```
Initialize fuel maps on the burning objects
for each time step do
    for each solid object do
        Diffuse temperature of the object
        Collect radiant heat from flame blobs and update temperature
        Generate flame blobs and update fuel density
    end
    for each gaseous fuel do
        Advect and diffuse the temperature and the density
        Collect radiant heat from flame blobs and update temperature
        Generate flame blobs and update density
    end for
    for each flame blob do
        Advect and diffuse density and temperature
        Generate smoke blobs
    end for
    for each smoke blob do
        advect and diffuse density of the smoke
    end for
end for
```

The diffusion of the temperature on the surface of each object is performed using a finite difference method on the subdivision of each surface. We have used a *Crank-Nicholson* scheme, because it is stable for any time step [77].

Specifically, the complexity of each time step depends only linearly on the number of blobs. By varying the rate and the size of the blobs, simulations with different accuracy and speed can be obtained. The blob representation also has the advantage that it can be mapped into several different rendering models. This enables the animator to choose between different depictions at different stages of the design process. For example, the coarsest depiction of a blob is to display only its centre. These different rendering models are outlined in the subsequent section.

6.5 Rendering

Realistic representation [...] depends not upon imitation or illusion or information but upon inculcation. Almost any picture may represent almost anything; that is, given picture and object there is usually a system of representation, a plan of correlation, under which the picture represents the object. *Nelson Goodman*

We have developed several blob rendering methods based on the techniques and algorithms explained in the previous chapter. The methods usually trade off visual quality for computational speed. Each algorithm is based on a approximation of the general transport equation. We now present these algorithms, starting from the coarsest approximation and moving on to more sophisticated renderings. But first, we provide an overview of the rendering parameters which are available to an animator.

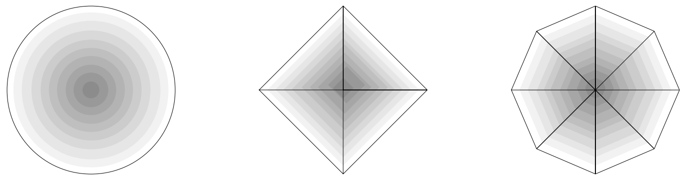

Figure 6.7: The transparency of a single blob (far right) can be approximated by projecting a set of translucent hardware shaded polygons.
of the transparency

6.5.1 Rendering Parameters

The animator controls the depiction of the density field by modifying the scattering, absorption and emissive properties of the medium. We have assumed that all these quantities are sampled only at three frequencies corresponding to the red, green and blue channels of a video display. These rendering characteristics depend on the density and on the temperature of the density field. Once the latter two fields are fixed, an animator can change the "look" of the phenomenon as follows. The scattering properties of the medium are entirely specified by the albedo and the phase function. The albedo gives the amount of scattering versus absorption for each wavelength and is a value between 0 (no scattering) and 1 (total scattering). The meaning is therefore fairly obvious to an animator. The phase function is characterized by its first moment $\bar{\mu} \in (-1, 1)$ and a factor $\mu_2 \in \{-1, +1\}$ accounting for total forward or backward scattering. An animator can model the anisotropy of the scattering using these parameters. The emission of the medium is controlled by the emission spectrum E, which multiplies the black body radiation given by the temperature (see Equation 5.1). Using this parameter, an animator can control the colour of the emission. The strength of the emission is given by the temperature. However, we do allow an animator to specify values bigger than 1 for the emission spectrum. The amount of transparency of the gas is a function of the extinction cross section σ_t and the density of the blob. The range of the extinction cross section extends from zero (totally transparent) to infinity (totally opaque). This range is not realistic for an animator to work with. In practice, we allow the animator control over the transparency τ (or equivalently the opacity) of a single blob. The extinction cross section is then recovered via:

$$\sigma_t = -\frac{1}{\bar{\rho}} \log(\tau),$$

where $\bar{\rho}$ is for example the average value of the density of the blobs. In our model there are, then, only 6 parameters which specify the rendering properties of the density field.

6.5.2 Real-Time Rendering of Blobs

In situations when each blob has the same illumination, the total transparency reaching the observer depends solely on the transparency of the blobs (see Equation 5.12). In this case, the transparency due to the blobs at a pixel $[x, y]$ on the screen is calculated by multiplying the transparencies due to each single blob:

$$\tau_{tot}[x,y] = \prod_{j=1}^{N} \tau_j[x,y].$$

For a given density and size, the transparency of a single blob can be precomputed and stored in a pixel map $\tau_j[x,y]$. In practice, each map can be obtained through an affine transformation from a single pixel map calculated for a particular smoothing kernel. This method was first introduced by Westover [103]. On machines which have fast polygon shading and alpha blending capabilities, we can apply polygons instead of the pixel maps [44]. Indeed, the projection of a spherical blob can be approximated by a triangular tesselation as shown in Figure 6.7. The alpha value of each vertex is set to the alpha value of the blob. The alpha value across each triangle is then computed by an interpolation of the graphics hardware. The rendering method is thus linear in the number of blobs, and produces very good results when the emission of the blobs is constant. Even when this is not the case, this approximation can be used to adjust the size and the transparency of the blobs in real time.

6.5.3 High Quality Rendering

Once the general appearance of the density field has been modelled using the real-time rendering approximation, high quality individual frames can be rendered for a final animation. We identify two different levels of rendering. In the first method only direct lighting is modelled. Hence we consider only one shooting operation from the patches and light sources to the blobs and one shooting operation from the blobs towards the surface patches. This approximation accounts for most visible light observed in real environments and is sufficient in most practical situations. For highly diffuse environments, or in the presence of thick scatterers, the multiple-scattering in the environment and in the density field can increase the realism by "brightening up" and "colour blending" parts of the scene. In both these algorithms, we can choose to ignore shadowing effects. In practice, we have assigned a flag to each object and density field indicating shadow calculations should be performed on or from them.

6.6 Interactive Modeller

We have combined each of the components into an interactive modeller. The program called "Whorl" allows a user to design motion fields and view their effect on blobs in real time. This modeller can output rendering scripts to a standard ray-tracer called "Optik" which we have modified to include the density rendering effects. We now explain both programs in more detail.

6.6.1 Whorl

Overview

Whorl allows a user to interactively design motion fields. We use the "Forms" [66] graphical user interface which we have modified to handle a slider with values that can be typed in manually. A particular field is designed as the superposition of instances of primitive fields. The instances are placed in the environment using translations, rotations and scaling. Furthermore, the magnitude and the decay of the field can be controlled. The user has a choice of precalculated turbulent motion fields with different eddie sizes and temporal decays. Once a turbulent field is loaded, its characteristics are modified using affine transformations as in the smooth field. The user can then either visualize the field as a two-dimensional slice of the wind fields, or can view its affect on moving blobs. The generation of the blobs is entirely specified by a region and a rate. The blobs are then generated uniformly on the region at the given rate. The animator also specifies the initial size and densities of the blobs. The evolution of the blobs is then controlled through the motion fields and by the diffusion and absorption rates. The appearance of the blobs is modified by varying the parameters described in Section 6.3.2. Motion fields only influence blob sets to which they are assigned. This allows motion fields to affect only certain blob sets in the environment. In addition to the blobs and the wind fields, we have included simple objects in Whorl such as spheres, cubes and cones. As in the smooth wind fields, each object is instantiated and transformed using affine transformations. In addition to its rendering parameters, each object has some parameters which characterize its burning properties. These are specific heat, heat diffusion and an initial fuel map. At each step of the simulation, the temperature on each object is updated and flame blobs are generated accordingly. A user starts a fire by lighting a "virtual match" somewhere in the environment using the mouse. Essentially all the parameters in Whorl can be modified using spline or linear interpolants. This is important when certain effects have to be choreographed. Each specification made by the user can be stored in a script, which can be further modified for fine-tuning a simulation.

Data Structures

Next we describe the data structures in C language of the main objects used in Whorl. We define types for three-dimensional vectors of floats (`VECTOR`), spectral colour samples (`COLOUR`) and four by four transformation matrices (`MATRIX`). The basic wind fields that we consider are entirely described by a transformation matrix, magnitude, drop off factor and a function which evaluates the field at a given location. Turbulent wind fields also need a pointer to a four dimensional table of vectors and a temporal scaling factor.

```
typedef
   struct
   {
      int type;             /* type of field */
      float magnitude;      /* magnitude of the field */
      float halfdist;       /* drop off factor */
      MATRIX m;             /* transformation matrix */
      float (*eval_func)(); /* field evaluation field */

/* --- following fields are only used by turbulent fields */
      int N;                /* size of grid defining the turbulence */
      float *****V;         /* vector data */
      t_scale;              /* temporal scaling */
/* --- following field is only used by ring vortex field */
      float radius;

   } FIELD_TYPE;
```

Each blob is defined by its position, age, and illumination parameters. There are two size parameters accounting for the different spreads of the density and the temperature of each blob.

```
typedef
   struct
   {
      float age;            /* time elapsed since creation */
      VECTOR pos;           /* position of the blob */
      float size_dens;      /* spread of the density */
      float size_temp;      /* spread of the temperature */
      float dens;           /* density at center of the blob */
      float temp;           /* temperature at the center of the blob */
/* --- illumination parameters:  */
      COLOUR extinction, emission, albedo;
   } BLOB_TYPE;
```

The characteristics of each set of blobs is defined by a function which generates new blobs at each time step, the parameters of the advection-diffusion equation and by a set of illumination parameters.

```
typedef
   struct
   {
      BLOB_TYPE * blobs;    /* pointer to the actual blobs */

/* --- blob generation parameters */
      int new_N;            /* number of new blobs to generate at each time step */
      float new_size;       /* initial spread */
      float new_dens;       /* initial density of each blob */
      float new_temp;       /* initial amount of temperature */
      int new_type;         /* type of region over which blobs are to be generated */
      MATRIX m;             /* transformation to be applied to the region */
      void (*new_func)();   /* function which generates new blobs */

/* --- blob evolution parameters */
      FIELD_TYPE * fields;  /* list of fields which affect the blobs */
      float diff_dens;      /* diffusion coefficient for the density */
      float diff_temp;      /* diffusion coefficient for the temperature */
      float abs_dens;       /* absorption coefficient for the density */
      float abs_temp;       /* absorption coefficient for the temperature */

/* --- initial blob illumination parameters */
```

```
      COLOUR extinction;    /* extinction cross section */
      COLOUR emission;      /* self-emission */
      COLOUR albedo;        /* albedo */

/* --- blob warp parameters */
      int warp_active;      /* flag indicating if warping is on */
      float delta_t;        /* used if the warping interval is kept constant */
      float int_size;       /* size of the integration step */
      float increase;       /* increase factor for sphere bounding warped blob */

/* --- following only used for flame blobs */
      FIRE_TYPE * fire;
   } BLOB_SET_TYPE;
```

The data structure for the objects are given by the type, transformation matrix and surface illumination and burning properties.

```
typedef
   struct
   {
      int type;             /* type of the object */
      MATRIX m;             /* transformation */
      SHADING_MODEL shade_prop; /* surface illumination parameters */
/* --- burning properties of the object */
      float diff_coeff;     /* fuel diffusion */
      float av_abs;         /* average absorption coefficient */
      float spec_heat;      /* specific heat of the object */
      int fuel_res;         /* resolution of the fuel map */
      float ** fuel_dens;   /* density of the fuel */
      float ** fuel_temp;   /* temperature of the fuel */
      float ** burnt_tex;   /* burnt texture map */
   } OBJECT_TYPE;
```

The data structure SHADING_MODEL depends on the type of shading model used (usually Phong's model). The fire data structure is created using two blob sets, one for the flame and one for the smoke.

```
typedef
   struct
   {
      OBJECT_TYPE * burn_objects; /* list of objects which are allowed to burn */
      float Ta;             /* activation temperature of flame */
      float Ts;             /* smoke creation temperature */
      float spec_heat;      /* specific heat */
      BLOB_SET_TYPE f;      /* density of flame */
      BLOB_SET_TYPE s;      /* density of smoke */
   } FIRE_TYPE;
```

In order to change the various parameters of these data structures over time we maintain a "change list" which contains pointers to data structures which are to be updated according to a given scheme at each time step. Each element in the change list is given by:

```
typedef
   struct
   {
      int n_change;           /* size of data to be changed (in floats) */
      float *change_data;     /* pointer to the data structure to be changed */
      void (*change_func)();  /* function used to change the data */
      void * data;            /* data that is used to do the change */
   } CHANGE_TYPE;
```

The last two fields in the data structure are dependent on the type of change. The three types supported by our system are constant value, linear interpolation and spline interpolation.

Code of Main Algorithms

The main loop of the modeller checks for events and handles them using callback routines from the "Forms" interface. The program is either in "simulation" mode or in "idle" mode. In the simulation mode time is incremented and all the primitives such as wind fields and blobs are updated.

```
main()
{
/* --- initialize variables/windows/interface */
   do_initialize_all ();
   time = 0.0;

/* --- main loop */
   for(;;)
   {
   /* --- check for user action via FORMs */
      obj = fl_check_forms ();
      if ( obj == quit_button ) break; /* user wants to stop */
      if ( obj == FL_EVENT ) handle_devices ();
   /* --- update scene */
      time += time_step;
      if ( simulation == IDLE ) continue;
      do_changes ( time );
      update_blob_sets ( time, time_step );
      update_fire ( time );
      update_view ( time );
   }
/* --- clean up at end of the program */
   cleanup ();
}
```

The function `do_changes()` updates the data structures pointed by each element in the change list of type `CHANGE_TYPE`. The function `update_blob_sets()` generates new blobs and updates the the properties of existing blobs.

```
void update_blob_sets ( float time, float time_step )
{
   BLOB_SET_TYPE *bs;

   for ( bs=head_blob_set_list ; bs ; bs++ )
   {
      bs->new_func ( bs, time_step ); /* create new blobs */
      update_blobs ( bs, time, time_step );
   }

   return;
}
```

The blob creation function for non-flame blob sets usually generates blobs uniformly within a given domain. For example, this domain could be a box. The transformation matrix `m` in the definition of the blob set gives the location, size and orientation of the domain. A typical function is given by the following code.

```
void new_func ( BLOB_SET_TYPE *bs, float time_step )
{
   BLOB_TYPE * b;
   int i;

   for ( i=0 ; i<bs->new_N ; i++ )
   {
      b = (BLOB_TYPE) get_memory (sizeof(b));
      b->pos = uniform_sample ( bs->new_type ); /* generate a sample location */
      b->pos = matrix_mult ( bs->m, b->pos ); /* transform */
      b->dens = time_step*bs->new_dens/bs->new_N;
      b->temp = time_step*bs->new_temp/bs->new_N;
      add_new_blob ( bs, b ); /* add blob to list */
   }

   return;
}
```

The function `update_blobs` moves and changes the properties of the blobs according to the algorithm of Section 4.3.2.

```
void update_blobs ( BLOB_SET_TYPE *bs, float time, float time_step )
{
   BLOB_TYPE *b;
   FIELD_TYPE *f;
   VECTOR vel;
   float volume;

   for ( b=bs->blobs ; b ; b++ )
   {
      /* --- advect blobs */
      for ( f=bs->fields ; f ; f++ )
      {
         vel = calc_velocity ( f, b->pos, time );
         b->pos += time_step * vel;
      }

      /* --- update other properties */
      b->age += time_step;
      b->size_dens = sqrt(bs->new_size**2+b->age*bs->diff_dens);
      b->size_temp = sqrt(bs->new_size**2+b->age*bs->diff_temp);
      volume = (bs->new_size/b->size_dens)**3;
      b->dens = bs->new_dens*exp(-bs->abs_dens*b->age)*volume;
      volume = (bs->new_size/b->size_temp)**3;
      b->temp = bs->new_temp*exp(-bs->abs_temp*b->age)*volume;
   }

   return;
}
```

The spread of the flame is computed by the following routine which updates the temperature maps of all the objects in the scene.

```
void update_fire ( float time_step )
{
   FIRE_TYPE * fi;
   OBJECT_LIST * obj;
   int i, j;
   VECTOR c_mass;
   float diff, c_size, c_power, form_fact, fuel_power;

   for ( fi = head_fire_list ; fi ; fi++ )
   {
   /* --- calculate average values of flame */
      c_mass = /* center of mass */
      c_size = /* average size */
      c_power = /* average power radiated */

   /* --- collect temperature of the flame on the objects */
      for ( obj=fi->burn_objects ; obj ; obj++ )
      {
         for ( i=0 ; i<obj->fuel_res ; i++ )
            for ( j=0 ; j<obj->fuel_res ; j++ )
```

```
          {
             form_fact = /* form factor between patch and center of mass of fire */
             fuel_power = /* σ * (obj->fuel_temp[i][j])**4 */
             diff = (form_fact*c_power - 2*PI*fuel_power)/obj->spec_heat;
             obj->fuel_temp += obj->fuel_dens[i][j]*obj->av_abs*diff;
          }
        /* --- diffuse temperature using Crank-Nicholson scheme */
        do_diffuse ( obj->fuel_temp, time_step );
      }
   }

   return;
}
```

The creation of flame and smoke blobs is done by defining appropriate routines pointed by the `new_func` field of the blob set type. These two functions are given next.

```
void new_flame_blobs ( BLOB_SET_TYPE *bs, float time_step )
{
   OBJECT_TYPE * obj;
   float rate;
   int i, j;

   for ( obj=bs->fire->objects ; obj ; obj++ )
   {
      for ( i=0 ; i<obj->fuel_res ; i++ )
         for ( j=0 ; j<obj->fuel_res ; j++ )
         {
            rate = exp(-bs->fire->Ta/obj->fuel_temp[i][j]);
            bs->new_dens = obj->fuel_dens[i][j]*rate;
            bs->new_temp = bs->fire->spec_heat * bs->fire->Ta*rate;
            new_func ( bs, time_step );
         }
   }
   return;
}

void new_smoke_blobs ( BLOB_SET_TYPE *bs, float time_step )
{
   BLOB_TYPE * b;
   float rate;

   for ( b=bs->fire->f.blobs ; b ; b++ )
   {
      rate = exp(-b->temp/bs->fire->Ts);
      bs->new_dens = b->dens*rate;
      bs->new_temp = 0.0; /* not important for smoke */
      new_func ( bs, time_step );
   }

   return;
}
```

6.6.2 Optik

Optik is a standard ray-tracing platform in existence in the Dynamic Graphics Project for many years. Code has been added, allowing for the ray tracing of gaseous phenomena (including fire). The code is called each time a ray returns with an illumination value. This value is then modified to account for absorption, scattering or emission by the density along the ray. The effects of multiple scattering and emission of light from the density into the environment are resolved by adding a "shooting" pass prior to the ray tracing. Essentially, then, only two new routines have to be provided to the ray tracer: one preprocessing routine and a routine which calculates the interactions of the density with a ray. These routines are described next at a high level.

Pseudo Code

The preprocessing routine first calculates the hierarchical representation of all the blobs in the scene using the algorithm of Section 4.2.2. Each pair of blobs are grouped together by a blob that bounds them. Also average emission, center of mass and illumination properties are kept at each level node of the tree. Once the tree is constructed the interactions of light between the density and the rest of the environment are accounted for using the algorithm of Section 5.5. In short:

Preprocess:

```
    Build hierarchical tree of blobs
    Calculate interactions with the environment
```

Once the intensity coming from the blobs has been precalculated the scene can be rendered from a given view point by a modified ray tracing algorithm. The intensity value returned by the ray tracer is then modified by the following algorithm.

Density Interactions:

```
    if shadow ray then
        return transparency alone
    end if

    Build list of blobs intersecting the ray (Section 5.3.2)
    Sort list from front to back
    Render list using the algorithm of Section 5.3.2
```

6.7 Results

6.7.1 Non-Reactive Gases

In the introduction of this thesis we gave a concrete example of the design of a visual simulation of smoke rising from a stack. This is just one instance of many gaseous phenomena which can be modelled in this fashion. Several other examples are discussed next. The exact values of the parameters used in some of the simulations are given in Appendix 6.C.

Steam Rising from a Coffee Cup

We used a single directional heat field to model the upward motion of the steam. The turbulence was chosen to be relatively slowly evolving over time. The blobs were generated uniformly on the surface of a disk corresponding to the surface of the "coffee". Figure 6.10 depicts a final rendered image.

Smokey Planet

We can use our model to generate "non-realistic" landscapes. In this example we used different colours for the smoke. The trails of smoke in Figure 6.11 were generated from disk shaped sources similarly to the coffee steam.

Cigarette Smoke

We generated two trails of smoke at the tip of the cigarette shown in Figure 6.12. We favoured "blueish" scattering in order to approximate the Rayleigh scattering from small dust particles. Notice both the shadow and the reflection of the smoke in the ashtray.

Interaction of Object with the Gas

We model the effect of moving objects on the gas by fitting a set of repulsive radial fields (see Section 6.3.1) around arbitrary objects. Although this is a gross approximation of the actual wind fields caused by moving objects, we did obtain convincing results with this simple technique. In Figure 6.13 we show four frames from an animation of a sphere passing through a wall of smoke. Notice also the shadowing of the sphere on the gas, which is obtained via our shooting algorithm.

Turbulent Morphing

Our models can also be used for special effects such as 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 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). Figure 6.14 shows four frames from the animation.

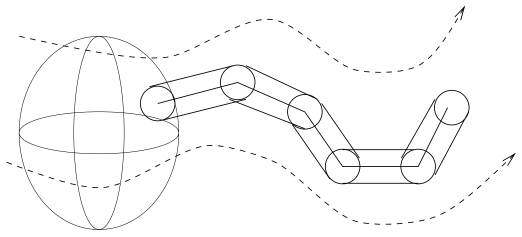

Figure 6.8: The hairs are modelled as trajectories of blobs through the wind fields.

6.7.2 Fire

Using the fire creation/spread model described in Section 6.4.2 a user can choreograph different burning processes. The spread is essentially controlled by a fuel map which can be painted onto objects in the scene. The spread is starting by applying a certain amount of heat somewhere in the environment. The simulation then commences and the user watches. Additional control is achieved through the rates of the reaction and the burning properties of the objects (diffusion rate/specific heat). It was found that it required some expertise to control the spread fire. However, we did manage to compute some convincing fire spreads. Some of these simulations are described next.

Smokeless Fire

Figure 6.16 shows a frame of an animation of the spread of fire and the burning of a structure made of "sticks". Notice how the fire naturally illuminates the environment through emission. This effect is calculated using our shooting algorithm described in Section 5.4. Figure 6.15 is another example.

Fire with Smoke

In Figure 6.17 we show four frames from an animation with the emission of dark smoke. We allocated a finite amount of fuel in the middle of the fire. The fire then extinguishes naturally when the amount of fuel decreases. Notice both the shadowing of the smoke and the emission due to the fire.

Multi-Coloured Fire

By playing around with the parameters of the fire model, we can generate flames emitting different colours. Figure 6.18 shows this. Notice also the very wispy smoke on the foreground which we have added for dramatic effect.

6.7.3 Hair

We model the motion and appearance of hair by tracing blobs with a fixed life-time through the motion field. The total history of the particle then constitutes one hair as shown in Figure 6.8. Each consecutive pair of blobs in the history define a single hair segment. The user can adjust the length and the resolution of the hairs by changing the life-time and the

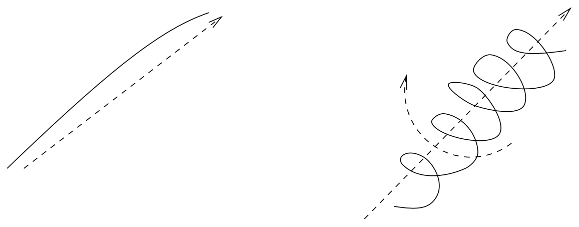

Figure 6.9: The curliness of each hair can be modelled by adding a twist.

number of blobs per hair respectively. Care has to be taken that the base blob of each hair is generated at the same location from frame to frame. This can be achieved by using the same seed for the random blob generator. Alternatively, the location and other parameters such as the colour of the hair can be texture mapped to allow a user to create a desired hair style or fur-pattern. The collisions of the hairs amongst themselves and with objects such as the head are accounted for naturally by the continuity of the motion fields. This enables a user to view complex simulations of hair in real time on a graphics work station. To add complexity to the hair we allow the user to perturb each hair individually. For example in order to create curly hair an additional twist can be added to each hair strand. This effect is depicted in Figure 6.9.

Adding dynamic behaviour to the hairs is often difficult using only motion field. We have resolved this problem by adding the effect of gravity by assigning a mass to each blob. Hence, thick straight hair exhibits more inertia than light curly hair, for example.

Hair Results

In the top of Figure 6.19 we show a rendering of a single hair. The hair on the right has a stronger twist. The two pictures on the bottom of the figure show the combination of many hairs emanating from a sphere. The picture on the right has self-shadowing of the hair turned on. In Figures 6.20 and 6.21 we show two pictures of "hairy creatures" created using the hair model. These last two pictures were created by Duncan Brinsmead and are included in this thesis with the permission from Alias/Wavefront.

Figure 6.10:

Figure 6.11:

Figure 6.12:

Figure 6.13:

Figure 6.14:

Figure 6.15:

Figure 6.16:

Figure 6.17:

Figure 6.18:

Figure 6.19:

Figure 6.20:

Figure 6.21:

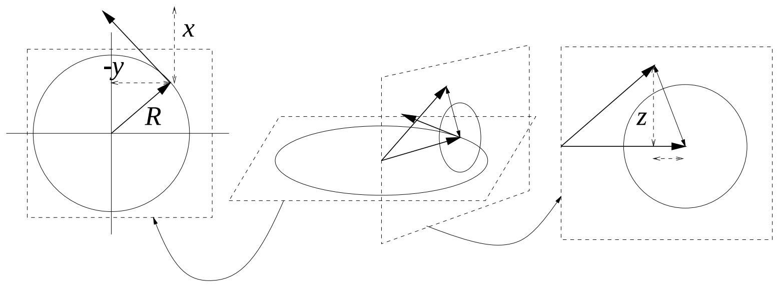

Figure 6.22: Vortex field around a circle.

Appendix 6.A Derivation of Vortex Fields

Very often a motion field is described by its vorticity field $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ instead. The equation for the evolution of the vorticity usually is simpler especially for two-dimensional motion fields. The velocity field can be recovered from the vorticity field through the integral [46]:[2]

$$\mathbf{u}_{vort}(\mathbf{x}, t) = -\frac{1}{4\pi}\int_{\mathbf{R}^3} \frac{(\mathbf{x}-\mathbf{x}') \times \boldsymbol{\omega}(\mathbf{x}', t)}{|\mathbf{x}-\mathbf{x}'|^3}\, d\mathbf{x}'.$$

This equation is then used to derive the velocity of the motion field from the equivalent vortex field. We will derive two motion fields taken from a vortex defined on a line and a circle respectively. If the vorticity is constant on the z-axis and pointed upward, then the corresponding velocity is given by:

$$\mathbf{u}_{vort}(\mathbf{x}) = -\frac{1}{4\pi}\int_{-\infty}^{\infty} \frac{(x, y, z-s) \times (0,0,1)}{(x^2+y^2+(z-s)^2)^{\frac{3}{2}}}\, ds = \frac{1}{2\pi}\frac{1}{x^2+y^2}(-y, x, 0).$$

To avoid the singularity near the z-axis and to localize the vortex field at the centre, we multiply the field by the smoothing function:

$$u_{lvort}(\mathbf{x}) = \frac{1}{2\pi}\frac{W(|\mathbf{x}|, \sigma)}{x^2+y^2}(-y, x, 0).$$

We now consider the case where the vorticity is defined by the vectors tangent to a circle of radius R lying in the xy plane and centred at the origin. For this vorticity, there is no closed form for the integral. We then approximate the integral by considering the contribution from the point on the circle closest to the point $\mathbf{x}$. This point can be calculated from geometrical arguments as depicted in Figure 6.22:

$$\mathbf{x}_c = (x_1, y_1, 0) = \frac{1}{\sqrt{x^2+y^2}}(xR, yR, 0),$$

and the vorticity at this point is given by the tangent to the circle:

$$\boldsymbol{\omega}(\mathbf{x}_c) = \frac{1}{\sqrt{x^2+y^2}}(-y, x, 0).$$

[2]Assuming no boundaries. In the presence of boundaries an additional term must be added to this equation.

The vorticity due to the circle is then given by

$$\mathbf{u}_{cvort}(\mathbf{x}) = -\frac{1}{4\pi}\frac{(\mathbf{x}-\mathbf{x}_c)\times(-y,x,0)}{\sqrt{x^2+y^2}|\mathbf{x}-\mathbf{x}_c|^3} = \frac{1}{4\pi}\frac{(zx,zy,(x_1-x)x+(y_1-y)y)}{\sqrt{x^2+y^2}((x-x_1)^2+(y-y_1)^2+z^2)^{3/2}}.$$

As for the line vortex, we multiply this velocity by a weighting function decaying with the distance from the circle.

Appendix 6.B Four-linear Interpolation of the Grid Values

Assume the vectors are stored in a the four dimensional grid $\mathbf{U}_{i,j,k,l}$. The spacing, orientation and location of the grid are entirely determined by the affine transformation $\mathbf{M}$. After multiplication by this matrix, we suppose that the origin of the grid is at the origin of the space, and has a spacing of one in each spatial dimension. After a scaling by the temporal spacing l_t, the temporal spacing can be assumed to be unity as well. Let (N_x, N_y, N_z, N_t) denote the respective resolutions of the grids in the corresponding coordinates. The algorithm then computes the velocity at a point (x, y, z, t) as follows:

```
$(i,j,k,l) = ([x],[y],[z],[t])$
$(a,b,c,d) = (x,y,z,t) - (i,j,k,l)$
$i = (i + N_x)\mathrm{mod}N_x$
$j = (j + N_y)\mathrm{mod}N_y$
$k = (l + N_z)\mathrm{mod}N_z$
$l = (l + N_t)\mathrm{mod}N_t$
$\mathbf{u}(x,y,z,t) = \mathrm{FourInterp}(\mathbf{U},i,j,k,l,a,b,c,d)$
```

In this algorithm "FourInterp" denotes the quadri-linear interpolation of 16 sample values to produce a continuous $\mathbf{u}(x, y, z, t)$.

Appendix 6.C Values of the parameters used in the simulations

In this section we give the exact parameters used in three of our simulations: "cigarette" (Figure 6.12), "coffee" (Figure 6.10) and "bonfire" (Figure 6.17). All turbulent wind fields were generated on a grid of size 16^4. Two different set of statistical parameters were used. The first one (Turb1) has $\epsilon = 1$, $\sigma = 1$ and $1/L = 4$. The second one (Turb2) has parameters $\epsilon = 1$, $\sigma = 1$ and $1/L = 2$. Since the cutoff wave number of Turb1 is higher than Turb2, Turb1 has smaller spatial structures. In each of the simulation there is a single directional wind field pointing in the "upward" y-direction. The values of the fields in each simulation are summarized in the following table.

simulation	field type	magnitude	half distance	scale	location
cigarette	Turb1	1	NA	1 1 1 5	0 0 0
	directional	1.5	60	1 1 1 NA	0 6 0
coffee	Turb2	2	NA	1 1 1 8	0 0 0
	directional	2	5000	1 1 1 NA	0 6 0
bonfire	Turb1	3	NA	3 3 3 20	0 0 0
	Turb1	1.5	NA	20 20 20 5	0 0 0
	directional	10	90	1 1 1 NA	0 -50 0

The generation of blobs in the cigarette and the coffee simulations are specified by the user. The definition of the source terms for both simulations is summarized in the following table.

simulation	`new_N`	`new_type`	`new_dens`	`new_size`	position	scale
cigarette	2	box	0.5	0.07	2.9 0 0	0.2 0 0.05
	2	box	0.5	0.07	2.9 4 0	0.2 0 0.05
disk	2	disk	0.5	0.1	0 6 0	4.5 4.5 4.5

The flame blobs and the smoke blobs in the bonfire example are generated by burning a rectangular object of size 20 by 20 located at location $(0, -35, 0)$ and oriented along the positive y-axis. The fuel map has a resolution of 20×20 and has an initial density of 1 and an initial temperature of 290. The specific heat of the fuel and the diffusion were both set to 1. The mean absorption cross section was set to 0.4. The parameters of the reaction of the creation of the smoke and the flame were: $T_a = 2300$, $T_s = 1500$, $\nu_a = 0.28$ and $\nu_b = 7$. The initial density and spread for the flame blobs and the smoke blobs are (0.05,7) and (0.03,10) respectively. The specific heat of the flame was set to 1.

Next we describe the parameters affecting the evolution of the blobs. In the simulation of the flame blobs we have set the diffusion of the density of the flame equal to the diffusion of the temperature of the flame.

simulation	diffusion	dissipation	time step
cigarette	0	0.005	0.1
coffee	0	0.05	0.1
bonfire (flame)	1	0.9	0.2
bonfire (smoke)	1.5	0.4	0.2

The illumination parameters of the simulation are given next. In each case the phase function is constant.

simulation	extinction	albedo	emission
cigarette	0.75 0.75 1.35	0.9 0.9 0.9	0 0 0
coffee	1.5 1.5 1.5	0.9 0.9 0.9	0 0 0
bonfire (flame)	0.34 0.09 0	0 0 0	black body
bonfire (smoke)	2 2 2	0.5 0.28 0.22	0 0 0

Blob warping is performed only in the bonfire simulation. Each flame blob was back-warped for a 4 seconds and each smoke blob for 2.5 seconds. The number of integration intervals were 9 and 4 for the flame and smoke blobs, respectively. The bounding radius was take to be 1.5 bigger than the unwarped radius for both type of blobs.

Chapter 7

Conclusions

The interval during which a painting is mistaken for the real thing, or a real thing for a painting, is the triumphant moment of trompe-l'oeil art. The artist appears to be as potent as nature, if not superior to it. Almost immediately, though, the spectator's uncertainty is eliminated by his recognition that the counterfeit is counterfeit. Once the illusion is dissolved, what is left is an object that is interesting not as a work of art but as a successful simulation of something that is not art. *Harold Rosenberg*

7.1 Summary of Results

In this thesis we have introduced several new methodologies for the visual simulation of natural phenomena. The method of separation of scales both in the design process and in the physical component of the simulation has proven to be effective. This was demonstrated for the large class of phenomena whose behaviour is subjected to a motion field. This is the case for both the density and the temperature fields of gaseous phenomena. The user controls the phenomenon through the motion field. This field is separated into a smooth and a turbulent component. The user designs the field by specifying the smooth component exactly and by providing the macroscopic statistical features of the turbulent component. We have developed new algorithms to generate turbulent components from its statistical description. Once the motion field is specified, it can be used to simulate many different phenomena. We have developed models for the evolution of density fields, temperature fields and hair strands under the influence of motion fields.

Stochastic models have been found to be successful in the simulation of small turbulent scales. The use of stochastic models is appropriate in case a physical simulation does not exist or is computationally too expensive. The derivation of proper stochastic models for natural phenomena is therefore an important part of a simulation. Very often a phenomenon A, for which a stochastic model is known, causes another phenomenon B. For example the wind causes ripples on the sea surface. We have outlined a general methodology to derive a stochastic model for B from the known stochastic model of the phenomenon A. An important example which we call *stochastic rendering* is to derive a stochastic model for the intensity field, given a stochastic model for the surface, density, light sources, etc. We have derived in particular a stochastic model for the intensity due to a density distribution.

We have outlined a general method of solving physical equations on an unordered set of points. This is preferable to algorithms which operate on a regular grid, especially for high dimensional problems. This methodology was used to derive efficient algorithms to solve advection-diffusion type equations. These equations were used to resolve the evolution of both the density and the temperature fields subjected to a wind field. The method has been used in the solution of the spatial variation of the intensity of light within a density distribution.

We have extended the global illumination solution to include light effects due to density distributions. In particular we have given a new method to solve for the effects of multiple scattering within the density distribution. The propagation due to multiple scattering is modelled as a diffusion process which can be derived directly from the scattering equation. The advantage of this method is that we get a set of equations which are local instead of the usual integral global equations of radiative transfer.

7.2 Future Research

The general methodology that we have proposed in this thesis has the potential of being applied to the simulation of many other phenomena. In particular the use of motion fields to control the simulation could be used to simulate water and the propagation of light in an environment. In the latter case this would allow artists to create specific illumination effects which are not necessarily realistic. One drawback of the use of motion fields is that they do not induce a dynamic behaviour automatically. Adding a dynamic component to the motion fields would greatly enhance the realism of the hair simulation and has the potential to be used for the realistic simulation of water. The use of stochastic models has proven to be very powerful in synthesizing complicated motions which are hard to create manually. Its use can be applied to many other phenomena. In particular the technique of stochastic derivation has not been explored to its fullest in this thesis. For example, stochastic rendering algorithms can be developed for the scattering from rough surfaces, the illumination from fluctuating light source and to simulate the caustics due to a random water surface. Another large area of application of the stochastic method is the motion of objects under the influence of a turbulent force. The general method of solving equations on an unordered set of data points can be refined and applied to many more areas of computer graphics and in other fields. For example the blob warping technique could be extended to blobby surfaces in order to model surfaces which are deformed by a surrounding field. The diffusion approximation to the propagation of light in a dense medium can be used to model subsurface multiple-scattering and glows from light sources.

Bibliography

[1] K. A. Anjyo, Y. Usami, and T. Kurihara. "A Simple Method For Extracting the Natural Beauty of Hair". *ACM Computer Graphics (SIGGRAPH '92)*, 26:111–120, July 1992.

[2] D. R. Baum, H. E. Rushmeier, and J. M. Winget. "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors". *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):325–334, July 1989.

[3] P. Blasi, B. Le Saec, and C. Schlick. "A Rendering Algorithm for Discrete Volume Density Objects". *Computer Graphics Forum*, 12(3):201–210, 1993.

[4] J. F. Blinn. "Simulation of Wrinkled Surfaces". *ACM Computer Graphics (SIGGRAPH '78)*, 12(3):286–292, 1978.

[5] J. F. Blinn. "A Generalization of Algebraic Surface Drawing". *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[6] J. F. Blinn. "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces". *ACM Computer Graphics (SIGGRAPH '82)*, 16(3):21–29, July 1982.

[7] J. P. Boon. *Lattice Gas Automata: A New Approach to the Simulation of Complex Flows*, pages 25–45. M. Mareschal, Plenum Press, New York, 1990.

[8] C. Bouville. "Bounding Ellipsoids for Ray-Fractal Intersection". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):45–52, July 1985.

[9] J. D. Buckmaster, editor. *Frontiers in Applied Mathematics. The Mathematics of Combustion.* SIAM, Philadelphia, 1985.

[10] S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. "A Progressive Multi-Pass Method for Global Illumination". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):165–174, July 1991.

[11] N. Chiba, K. Muraoka, H. Takahashi, and M. Miura. "Two-dimensional Visual Simulation of Flames, Smoke and the Spread of Fire". *The Journal of Visualization and Computer Animation*, 5:37–53, 1994.

[12] A. J. Chorin. *Vorticity and Turbulence.* Springer Verlag, New York, 1994.

[13] P. Clavin and F. A. Williams. "Theory of Premixed-Flame Propagation in Large-Scale Turbulence". *The Journal of Fluid Mechanics*, 90(3):589–604, 1979.

[14] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. "A Progressive Refinement Approach to Fast Radiosity Image Generation". *ACM Computer Graphics (SIGGRAPH '88)*, 22(4):75–84, August 1988.

[15] R. A. Drebin, L. Carpenter, and P. Hanrahan. "Volume Rendering". *ACM Computer Graphics (SIGGRAPH '88)*, 22(4):65–74, August 1988.

[16] J. J. Duderstadt and W. R. Martin. *Transport Theory.* John Wiley and Sons, New York, 1979.

[17] 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.

[18] A. Fournier. While chairing a session at SIGGRAPH'94.

[19] A. Fournier and W. T. Reeves. "A Simple Model of Ocean Waves". *ACM Computer Graphics (SIGGRAPH '86)*, 20(4):75–84, August 1986.

[20] G. Y. Gardner. "Visual Simulation of Clouds". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):297–384, July 1985.

[21] J. I. Gikhman and A. V. Skorohod. *The Theory of Stochastic Processes. Volumes I, II and III.* Springer Verlag, Berlin, Heidelberg, 1974-79.

[22] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: Theory and application to non spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[23] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Bataille. "Modeling the Interaction of Light Between Diffuse Surfaces". *ACM Computer Graphics (SIGGRAPH '84)*, 18(3):213–222, July 1984.

[24] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* The MIT Press, Cambridge, Massachusetts, 1988.

[25] W. Hackbusch. *Multi-grid Methods and Applications.* Springer Verlag, Berlin, 1985.

[26] P. Hanrahan and W. Krueger. "Reflection from Layered Surfaces due to Subsurface Scattering". In *Proceedings of SIGGRAPH '93*, pages 165–174. Addison-Wesley Publishing Company, August 1993.

[27] X. D. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg. "A Comprehensive Physical Model for Light Reflection". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):175–186, July 1991.

[28] R. W. Hockney and J. W. Eastwood. *Computer Simulation using Particles.* McGraw-Hill Inc., New York, 1988.

[29] D. Immel, M. Cohen, and D. P. Greenberg. "A Radiosity Method for Non-Diffuse Environments". *ACM Computer Graphics (SIGGRAPH '86)*, 20(4):133–142, August 1986.

[30] M. Inakage. “Volume Tracing of Atmospheric Environments”. *The Visual Computer*, 7:104–113, 1991.

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

[32] A. G. Journel and C. J. Huijbregts. *Mining Geostatistics.* Academic Press, New York, 1978.

[33] J. T. Kajiya. “The Rendering Equation”. *ACM Computer Graphics (SIGGRAPH ’86)*, 20(4):143–150, August 1986.

[34] J. T. Kajiya and T. L. Kay. “Rendering Fur with Three Dimensional Textures”. *ACM Computer Graphics (SIGGRAPH ’89)*, 23(3):271–280, July 1989.

[35] J. T. Kajiya and B. P. von Herzen. “Ray Tracing Volume Densities”. *ACM Computer Graphics (SIGGRAPH ’84)*, 18(3):165–174, July 1984.

[36] K. Kaneda, T. Okamoto, E. Nakamae, and T. Nishita. “Photorealistic Image Synthesis for Outdoor Scenery under Various Atmospheric Conditions”. *The Visual Computer*, 7:247–258, 1991.

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

[38] R. V. Klassen. “Modeling the Effect of the Atmosphere on Light”. *ACM Transactions on Graphics*, 6(3):215–237, July 1987.

[39] T. Koga. *Introduction to Kinetic Theory. Stochastic Processes in Gaseous Systems.* Pergamon Press, Oxford, 1970.

[40] R. H. Kraichnan. “Diffusion by a Random Velocity Field”. *The Physics of Fluids*, 13(1):22–31, 1970.

[41] P. Krée and C. Soize. *Mathematics of Random Phenomena: Random Vibrations of Mechanical Structures.* D. Reidel Publishing Company, Dordrecht, 1986.

[42] W. Krueger. “Intensity Fluctuations and Natural Texturing”. *ACM Computer Graphics (SIGGRAPH ’88)*, 22(4):213–220, August 1988.

[43] E. Languénou, K.Bouatouch, and M.Chelle. Global illumination in presence of participating media with general properties. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 69–85, Darmstadt, Germany, June 1994.

[44] D. Laur and P. Hanarahan. “Hierarchical Splatting: A Prograssive Refinement Algorithm for Volume Rendering”. *ACM Computer Graphics (SIGGRAPH ’91)*, 25(4):285–288, July 1991.

[45] A. M. LeBlanc, R. Turner, and D. Thalmann. “Rendering Hair using Pixel Blending and Shadow Buffers”. *The Journal of Visualization and Computer Animation*, 2:92–97, 1991.

[46] A. Leonard. "Vortex Methods for Flow Simulation". *Journal of Computational Physics*, 37:289–335, 1980.

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

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

[49] J. P. Lewis. "Algorithms for Solid Noise Synthesis". *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):263–270, July 1989.

[50] B. Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman and Co., New York, 1982.

[51] G. A. Mastin, P. A. Watterberg, and J. F. Mareda. "Fourier Synthesis of Ocean Scenes". *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.

[52] N. Max. "Vectorized Procedural Models for Natural Terrains: Waves and Islands in the Sunset". *ACM Computer Graphics (SIGGRAPH '81)*, 15(3):317–324, August 1981.

[53] N. Max. "Atmospheric Illumination and Shadows". *ACM Computer Graphics (SIGGRAPH '86)*, 20(4):117–124, August 1986.

[54] N. Max. "Light Diffusion through Clouds and Haze". *Computer Vision, Graphics, and Image Processing*, 33:280–292, 1986.

[55] N. Max. Efficient light propagation for multiple anisotropic volume scattering. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 87–104, Darmstadt, Germany, June 1994.

[56] 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.

[57] G. W. Meyer, H. E. Rushmeier, M. F. Cohen, D. P. Greenberg, and K. E. Torrance. "An Experimental Evaluation of Computer Graphics Imagery". *ACM Transactions on Graphics*, 5(1):30–50, January 1986.

[58] G. Miller and A. Pearce. "Globular Dynamics: A Connected Particle System for Animating Viscous Fluids". In *SIGGRAPH '89, Course 30 Notes: Topics in Physically-Based Modelling.* SIGGRAPH, Boston Massachusetts, August 1989.

[59] J. J. Monaghan. "Why Particle Methods Work". *SIAM Journal of Scientific and Statistical Computing*, 3(4):422–433, December 1982.

[60] J. J. Monaghan. "Smoothed Particle Hydrodynamics". *Annual Reviews of Astronomy and Astrophysics*, 30:543–574, 1992.

[61] J. J. Monaghan. "Simulating Free Surface Flows with SPH". *Journal of Computational Physics*, 110(2):399–406, 1994.

[62] A. S. Monin and A. M. Yaglom. *Statistical Fluid Mechanics.* The MIT Press, Cambridge, Massachusetts, 1975.

[63] T. Nishita, Y. Miyawaki, and E. Nakamae. "A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources". *ACM Computer Graphics (SIGGRAPH '87)*, 21(4):303–310, July 1987.

[64] D. H. Norrie and G. de Vries. *The Finite Element Method. Fundamentals and Applications.* Academic Press, New York, 1973.

[65] J. F. O'Brien and J. K. Hodgins. "Dynamic Simulation of Splashing Fluids". *Proceedings of Computer Animation '95*, April 1995.

[66] M. Overmars. Forms Library. A Graphical User Interface Toolkit for Silicon Graphics Workstations. Public domain software, 1992.

[67] S. Panchev. *Random Functions and Turbulence.* Pergamon Press, Oxford, 1971.

[68] Paramount. *Star Trek II: The Wrath of Khan.* Paramount Pictures, 1982.

[69] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.

[70] S. N. Pattanaik and S. P. Mudur. Computation of global illumination in a participating medium by monte carlo simulation. *The Journal of Visualization and Computer Animation*, 4(3):133–152, July–September 1993.

[71] D. R. Peachy. "Solid Texturing of Complex Surfaces". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):279–286, July 1985.

[72] K. Perlin. "An Image Synthesizer". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):287–296, July 1985.

[73] K. Perlin and E. M. Hoffert. "Hypertexture". *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):253–262, July 1989.

[74] C. H. Perry and R. W. Picard. "Synthesizing Flames and their Spread". *SIGGRAPH'94 Technical Sketches Notes*, July 1994.

[75] B. Phong. "Illumination for Computer Generated Pictures". *Communications of the ACM*, 18(6):311–317, 1975.

[76] T. Porter and T. Duff. "Compositing Digital Images". *ACM Computer Graphics (SIGGRAPH '84)*, 18(3):253–259, July 1984.

[77] 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.

[78] W. T. Reeves. "Particle Systems. A Technique for Modeling a Class of Fuzzy Objects". *ACM Computer Graphics (SIGGRAPH '83)*, 17(3):359–376, July 1983.

[79] 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.

[80] H. P. Robertson. “The Invariant Theory of Isotropic Turbulence”. *Proceedings of the Cambridge Philosophical Society*, 36(2):209–223, 1940.

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

[82] R. E. Rosenblum, W. E. Carlson, and E. Tripp III. “Simulating the Structure and Dynamics of Human Hair: Modelling, Rendering and Animation”. *The Journal of Visualization and Computer Animation*, 2:141–148, 1991.

[83] H. E. Rushmeier and K. E. Torrance. “The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium”. *ACM Computer Graphics (SIGGRAPH ’87)*, 21(4):293–302, July 1987.

[84] G. Sakas. “Fast Rendering of Arbitrary Distributed Volume Densities”. In F. H. Post and W. Barth, editors, *Proceedings of EUROGRAPHICS ’90*, pages 519–530. Elsevier Science Publishers B.V. (North-Holland), September 1990.

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

[86] G. Sakas and M. Gerth. “Sampling and Anti-Aliasing of Discrete 3-D Volume Density Textures”. In F. H. Post and W. Barth, editors, *Proceedings of EUROGRAPHICS ’91*, pages 87–102. Elsevier Science Publishers B.V. (North-Holland), September 1991.

[87] M. Shinozuka. “Simulation of Multivariate and Multidimensional Random Processes”. *The Journal of the Accoustical Society of America*, 49:357–367, 1971.

[88] M. Shinozuka and C. M. Jan. “Digital Simulation of Random Processes and its Applications”. *Journal of Sound and Vibration*, 25(1):111–128, 1972.

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

[90] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer.* Hemisphere Publishing Corp., Washington DC, 1981.

[91] F. Sillion and C. Puech. “A General Two-Pass Method Integrating Specular and Diffuse Reflection”. *ACM Computer Graphics (SIGGRAPH ’89)*, 23(4):335–344, August 1989.

[92] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. “A Global Illumination Solution for General Reflectance Distributions”. *ACM Computer Graphics (SIGGRAPH ’91)*, 25(4):187–196, July 1991.

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

[94] K. Sims. "Artificial Evolution for Computer Graphics". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):319–328, July 1991.

[95] D. Tonnesen. "Modelling Liquids and Solids Using Thermal Particles". In *Proceedings of Graphics Interface '91*, pages 255–262, June 1991.

[96] K. E. Torrance and E. M. Sparrow. "Theory for Off-Specular Reflection From Roughened Surfaces". *Journal of the Optical Society of America*, 57(9):1105–1114, September 1967.

[97] Roy Troutman and Nelson L. Max. Radiosity algorithms using higher order finite element methods. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 209–212, 1993.

[98] C. Upson and M. Keeler. "V-BUFFER: Visible Volume Rendering". *ACM Computer Graphics (SIGGRAPH '88)*, 22(4):59–64, August 1988.

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

[100] J. R. Wallace, M. F. Cohen, and D. P. Greenberg. "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Techniques". *ACM Computer Graphics (SIGGRAPH '87)*, 21(4):311–320, July 1987.

[101] J. R. Wallace, K. E. Elmquist, and E. A. Haines. "A Ray Tracing Algorithm for Progressive Radiosity". *ACM Computer Graphics (SIGGRAPH '89)*, 23(3):315–324, July 1989.

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

[103] L. Westover. "Footprint Evaluation for Volume Rendering". *ACM Computer Graohics (SIGGRAPH'90)*, 24(4):367–376, August 1990.

[104] H. Weyl. *Classical Groups. Their Invariants and Representations.* Princeton University Press, Princeton, New Jersey, 1946.

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

[106] A. M. Yaglom. *Correlation Theory of Stationary and Related Random Functions I. Basic Results.* Springer-Verlag, 1986.

[107] Harold R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 213–220, 1993.

[108] A. H. Zemanian. *Distribution Theory and Transform Analysis: An Introduction to Generalized Functions, with Applications.* McGraw-Hill, New York, 1965.