VISUAL MODELING AND SIMULATION OF MULTISCALE PHENOMENA

Rahul Narain

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2011

Approved by:

Ming C. Lin

Dinesh Manocha

Anselmo Lastra

Michael L. Minion

Theodore Kim

ABSTRACT

RAHUL NARAIN: Visual Modeling and Simulation of Multiscale Phenomena

(Under the direction of Ming C. Lin)

Many large-scale systems such as human crowds, fluids, and granular materials exhibit complicated motion at many different scales, from a characteristic global behavior to important small-scale detail. Such multiscale systems are computationally expensive for traditional simulation techniques to capture over the full range of scales. I present novel techniques for scalable simulation of these large, complex phenomena for visual computing applications. These techniques achieve their efficiency by coupling together separate models for the large-scale and fine-scale dynamics of a complex system.

In fluid simulation, it remains a challenge to efficiently simulate fine details such as foam, ripples, and turbulence without compromising the accuracy of the large-scale flow. I present two techniques for this problem that combine physically-based numerical simulation for the global flow with efficient local models for detail. For surface features, I propose the use of texture synthesis, guided by the physical characteristics of the macroscopic flow. For turbulence in the fluid motion itself, I present a technique that tracks the transfer of energy from the mean flow to the turbulent fluctuations and synthesizes these fluctuations procedurally, allowing extremely efficient visual simulation of turbulent fluids.

Another large class of problems which are not easily handled by traditional approaches is the simulation of very large aggregates of discrete entities, such as dense pedestrian

crowds and granular materials. I present a technique for crowd simulation that couples a discrete model of individual navigation with a novel continuum formulation for the collective motion of pedestrians, enabling simulation of extremely large crowds at near-real-time rates on commodity hardware. I also present a technique for simulating granular materials which generalizes this model and introduces a novel computational scheme for friction, thus efficiently reproducing a wide range of granular behavior.

In all these cases, the proposed techniques are typically an order of magnitude faster than comparable existing methods. Through these applications to a diverse set of challenging simulation problems, I demonstrate that the proposed approach is a powerful and versatile technique for the simulation of a broad range of large, complex systems.

ACKNOWLEDGEMENTS

There are many people who have helped make this dissertation possible, and I owe my gratitude to all of them.

First and foremost, of course, my advisor, Ming Lin, for all of the guidance and support she has given me throughout my years in graduate school. It is in no small part thanks to her constant encouragement, feedback, and patience that I have been able to grow the seeds of my ideas to fruition in their current form in this dissertation.

I am also grateful to all of my committee members: Dinesh Manocha for always prodding us in GAMMA to do more and do better; Anselmo Lastra for many things, but especially for his advice that made me a better public speaker; Michael Minion for making absolutely sure I knew the math inside and out; and Ted Kim for asking just the right questions, thus compelling me to strengthen the weakest parts of my thesis.

To my colleagues and friends in the GAMMA group, I am grateful for all the incisive feedback and the stimulating discussions we had in the weekly meetings. Abhinav Golas, Jason Sewall, Sean Curtis, Vivek Kwatra, and Huai-Ping Lee especially have my thanks for sharing the pain of working on a paper submission deadline. So long, and thanks for all the foosball.

Last, but certainly not least, I must thank my parents for giving me the wisdom to choose the path I wanted to take in life and the freedom to take it. From them I have received nothing but encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In nature and in society, we are surrounded by numerous large-scale phenomena of remarkable beauty and complexity. The chaotic, swirling motion of vortices in turbulent fluids; the bustling flow of pedestrian crowds; and the sometimes stiff, sometimes fluid behavior of sand and other granular materials; these are just a few examples of systems that show many levels of complex behaviour. The dynamics of such systems can be viewed on a wide range of scales: from the chaotic, fluctuating interactions between individual objects on the finest scales, to the coherent aggregate flow of the system on the largest scales. This interplay between the bulk motion and the fine detail in these so-called *multiscale phenomena* makes them extremely rich and fascinating from a theoretical as well as a visual perspective.

One of the primary driving forces in computer graphics is the desire to reproduce the beauty and richness of the real world as closely as possible in virtual scenes. Modeling the visually striking behavior of multiscale phenomena is an important part of this goal for realistic and compelling graphical applications. However, the wide range of scales over which the dynamics of such phenomena take place makes them very challenging to simulate faithfully.

For example, consider the swirling, ever-changing flow of a turbulent river (figure 1.1), which exhibits patterns on a wide range of scales. At the coarsest level of abstraction, on a scale of tens and hundreds of meters, the river has an overall flow in the downstream

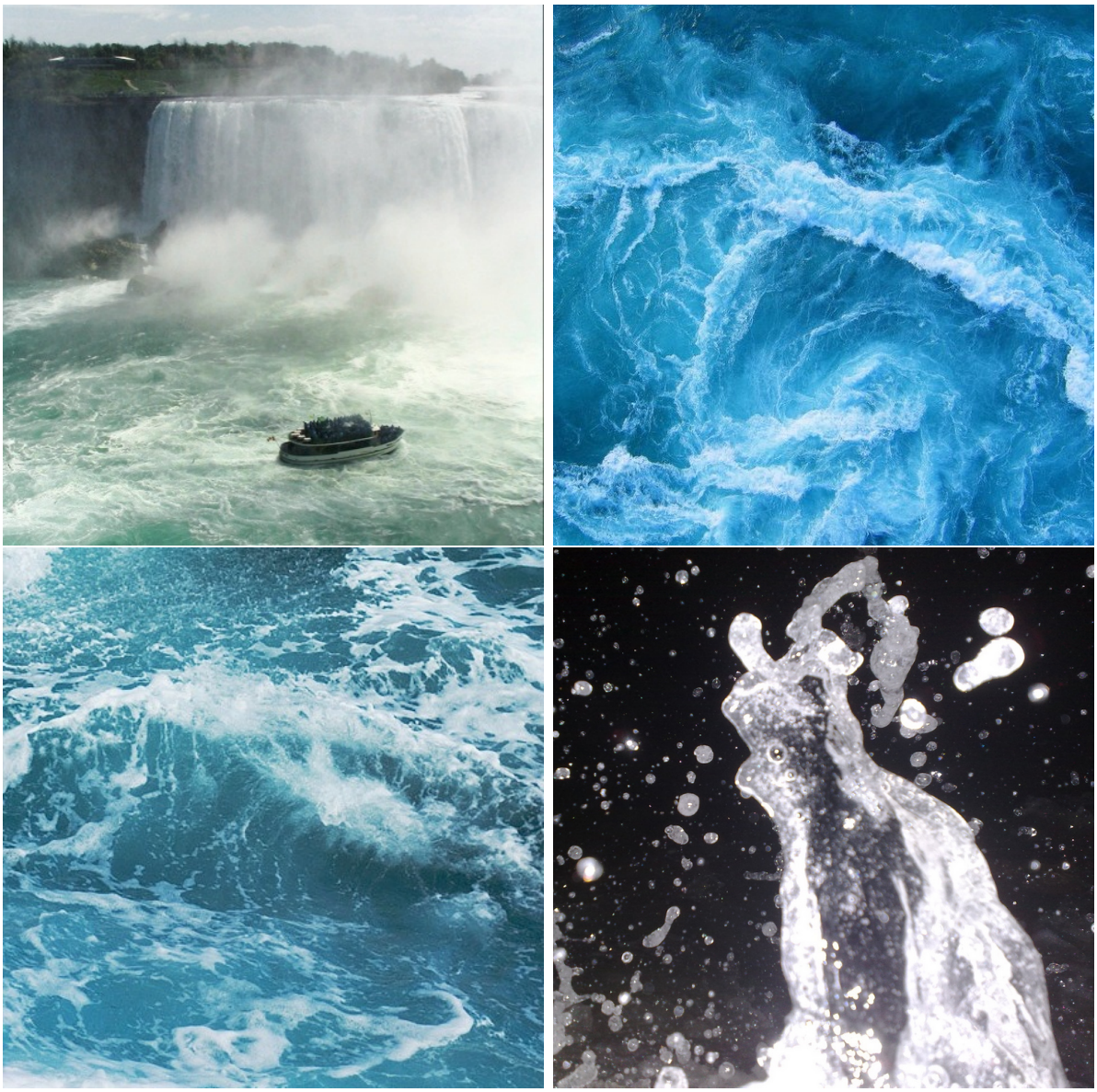


Figure 1.1: The many scales of motion in a turbulent river, (in reading order) from an overall downstream flow, through large waves and smaller splashes to the smallest individual droplets and bubbles. (*Image credits: "niagara falls #0007" by Flickr user workinpana, "Turbulence 3" by mvisosky, "Turquoise is the color" by Josh Schwartzman, "Ocean Spray" by James Nash, all licensed under Creative Commons.*)

direction, driven by the force of gravity. But this downstream flow is broken up by large disturbances such as whirlpools and waves caused by the interaction of the fluid with obstacles in its path. Interactions between these features gives rise to smaller and smaller vortices and eddies that fluctuate chaotically, giving the flow its characteristic turbulent appearance. This process goes on to the smallest scales, with small ripples and bubbles on the order of millimeters producing foam and spray on the water's surface. All of these diverse characteristics of motion spanning several orders of magnitude in spatial scales together form the flow of this turbulent fluid, and it is the combination of features at all these different scales that give the scene its visual richness and complexity.

Similar characteristics can be observed in many other large-scale systems. In a large, dense crowd of pedestrians, for example, each individual plans his or her own path independently, navigating around neighboring people and obstacles in a complicated way. From a bird's eye view, on the other hand, the crowd's behaviour appears as a coherent flow of pedestrian traffic which shows macroscopic fluid-like effects like lane formation, congestion at narrow passages, and an overall bulk flow of the crowd as a whole. The flow of granular materials such as sand shares many of these properties. While such a material is composed of millions of discrete, rigid grains in close frictional contact with each other, at a macroscopic scale the material as a whole flows smoothly and conforms to the shape of its container, as if it were a fluid. More examples include biological structures such as skin and muscle, cloth, hair, piles of objects, and many others. Indeed, in principle almost any system can be thought of in a multiscale way, since it is inevitably composed of smaller interacting structures on a finer scale. However, in this dissertation, I restrict attention to those systems where the many of these scales show complex, visible dynamics that need to be captured for computer animation applications.

It remains a challenge to efficiently model and simulate such large, complex systems in a way that can faithfully capture their diverse behaviors over a wide range of scales, including both the coherent large-scale flow and complex fine-scale detail. These phenomena turn out

to be very expensive to model using traditional computer graphics techniques, ultimately due to the difficulty of representing both the finest and the coarsest scales in a single mathematical model. To make practical and efficient simulation possible, it is necessary to develop a new class of techniques designed for such multiscale phenomena. Such techniques must couple together different mathematical models for the dynamics of the system at different scales, forming a heterogeneous representation that captures all the behaviors of interest at a greatly reduced computational cost.

1.1 MULTISCALE PHENOMENA

As we have seen, many real-world phenomena show complex behavior over a wide range of spatial scales. In these multiscale phenomena, the large and small scales can be termed as "macroscopic" and "microscopic", respectively, although this distinction should be seen as describing two directions on a continuum of scales, rather than a strict dichotomy. Typically, features at differing scales behave in qualitatively different ways, yielding a rich variety of behaviors.

At the smallest scales, the dynamics of local features is typically very complex. The basic entities forming the system often interact in ways that are stiff, nonlinear, and chaotic. For example, the contact and friction forces between closely packed grains in a sand pile produce a strongly linked network of coupled interactions that transmits forces over large distances through the bulk of the material. Pedestrians in a dense crowd continually adjust their velocities with respect to the motion of their neighbors in a nonlinear way that is hard to simplify directly. Finally, the behavior of vortices in a turbulent fluid is a classical example of a highly chaotic, nonlinear, and unpredictable system, which has resisted theoretical analysis for over a century. In all such cases, the detailed behavior of such systems often exhibits either rapid, stochastic fluctuations in both space and time ("rapid" being relative to the large-scale evolution of the system), and/or very tightly coupled dynamics across long chains of neighborhoods.

Nevertheless, despite the complexity of these local interactions, on the large scale their bulk effect is to give rise to a coherent high-level flow at the macroscopic level. That is, the fluctuations in the local interactions can typically be abstracted away to reveal a smooth macroscopic behavior that is coherent over both space and time: the overall global motion of a fluid, of a crowd of pedestrians, or of a pile of sand. This large-scale flow is dependent not so much on the precise microscopic state of the system, but more on its averaged statistical characteristics—the density of entities, large-scale obstacles, bulk forces, and so on. Therefore, one can expect that this motion can be described well using a simpler macroscopic model.

The dynamics at different scales are however not entirely independent. On a foundational level, it is after all the nature of interactions at the fine scales that determines the characteristics of the macroscopic behavior. More practically, even when this macroscopic flow is modeled directly at the coarse scale, such a model typically cannot be "closed" without reference to properties such as the distribution of features that can only be resolved at the microscopic level. Conversely, the local interactions between individual entities are heavily influenced by the large-scale context within which the entities are embedded. This two-way interaction between scales is an important part of the behavior of complex systems.

The specific problem I address in this thesis is that of developing efficient techniques for visual modeling and simulation of such multiscale phenomena for realistic graphics and animation. For such applications, it is necessary to model the behavior at both large and small scales. As the small-scale dynamics of a multiscale phenomenon are visually perceived at a largely qualitative level, it is not exact accuracy that is desired at the microscopic level, but rather faithful reproduction of qualitative, statistical features. On the other hand, as the interactions of coherent large-scale features are readily observable, deviation from the true behavior is often easily perceived. Therefore, the macroscopic behavior of the system must be accurately modeled in a way that is physically or

quantitatively sound.

1.2 VISUAL MODELING AND SIMULATION

In the computer graphics literature, numerous techniques have been proposed for modeling a wide range of natural and social phenomena. Three broad classes of techniques that are relevant to the current topic are discussed below: procedural models, data-driven techniques, and direct numerical simulation.

Procedural models take a largely phenomenological approach, using simple procedures to emulate the visual appearance of a given system, without reference to the underlying dynamics. Many such methods have been proposed using various techniques such as Fourier synthesis (Fournier and Reeves, 1986; Shinya and Fournier, 1992; Stam and Fiume, 1993), random noise functions (Perlin, 1985; Perlin and Neyret, 2001; Perlin, 2002; Kniss and Hart, 2004; Cook and DeRose, 2005; Bridson et al., 2007), particle systems (Sims, 1990), and flow fields (Chenney, 2004). These include some of the earliest and simplest techniques for modeling physical phenomena. While these often work very well to model the stochastic fine-scale features of a complex system, they can only be designed for very specific scenarios and cannot be extended to a general context. In particular, since such models have no notion of the underlying physical dynamics of the system, they cannot respond to interactions with the environment or with external forces, making them inadequate for modeling large-scale flows.

There are a number of data-driven techniques which build a non-parametric model from a large number of observations, whether from real-world experiments or through precomputation of high-resolution simulations. These are used extensively in material models for rendering (Ward, 1992; Marschner et al., 1999; Matusik et al., 2003), character animation (Arikan and Forsyth, 2002; Kovar et al., 2002; Kovar and Gleicher, 2004; Safonova and Hodgins, 2007), and example-based texture synthesis (Efros and Leung, 1999; Ashikhmin, 2001; Efros and Freeman, 2001; Kwatra et al., 2005; Lefebvre and

Hoppe, 2006). For physically based simulation, data-driven techniques have been applied to many problems such as cloth (Cordier and Magnenat-Thalmann, 2005; de Aguiar et al., 2010), fluids (Treuille et al., 2006b; Wicke et al., 2009), nonlinear materials (Barbič and James, 2005; An et al., 2008) and many others. One pervasive challenge for data-driven techniques in animation is the high dimensionality of the state space of most complex systems, which necessitates the precomputation, storage, and lookup of an extremely large database of samples, and also limits the results to lie within the span of existing observations. This is especially a problem for multiscale systems, which have exponentially many degrees of freedom in the fine scales.

The most general technique for modeling natural phenomena is that of direct numerical simulation. Here, one directly solves the underlying equations that govern the behavior of the system of interest, enabling all the relevant dynamics of the system to be captured in general scenarios. Due to the power and flexibility of this approach, it is the *de facto* standard for modeling many complex physical phenomena in computer graphics. For example, fluids such as smoke and water can be animated by numerically solving the Navier-Stokes equations, and a number of numerical methods have been proposed using different discretizations such as rectilinear grids (Foster and Metaxas, 1996; Stam, 1999; Foster and Fedkiw, 2001; Bargteil et al., 2006a), tetrahedral meshes (Feldman et al., 2005; Klingner et al., 2006; Chentanez et al., 2007), particles (Müller et al., 2003; Premoze et al., 2003), and vorticity representations (Angelidis and Neyret, 2005; Park and Kim, 2005; Elcott et al., 2007). For various other phenomena as well, there is a vast literature of techniques for numerical simulation, which is too broad to cover here; some key works will be cited when relevant in the following chapters.

The power of numerical simulation, however, comes at a high computational cost, especially for multiscale phenomena. To capture all the small-scale details, the discretization of the numerical scheme must be fine enough to resolve the smallest features of interest, while the size of the domain and the macroscopic features is far larger. Therefore, an

enormous number of computational elements are needed, along with a correspondingly large memory requirement and computation time, both of which rise steeply as the resolution of the simulation is refined. For example, to simulate a given volume of fluid numerically for unit time, the memory requirement rises as $\Omega(n^3)$ and the compute time as $\Omega(n^4)$, where the simulation is resolved to n elements per linear dimension. This makes it prohibitively expensive to perform high-resolution simulation of fluids using purely numerical techniques. This problem of scale is a common challenge faced by numerical simulation of all multiscale phenomena.

Each of the above approaches is in itself inadequate for modeling multiscale phenomena. Procedural methods can convincingly model the chaotic fine-scale behavior, but cannot reproduce the global flow with large-scale physical interactions. Data-driven techniques are challenged by the high number of degrees of freedom at fine scales. Numerical simulation can capture all scales of dynamics but at a prohibitive computational cost, rendering it practical only for coarsely resolved large-scale flow. However, as each of these techniques are often well-suited for particular scales of behavior with qualitatively different properties, a combination of different such models has the potential to efficiently capture both large and small scales, making it practical for the first time to perform highly detailed, realistic simulation of multiscale phenomena for visual applications. This is the key idea upon which my thesis is based.

1.2.1 MULTISCALE METHODS

In applied mathematics, there have been many recent developments in methods for solving problems that possess multiple scales. These are broadly termed *multiscale methods*, and they rely on a similar insight of combining multiple models and representations of a multiscale system. The field of multiscale methods is extremely broad and covers many kinds of techniques for different classes of phenomena; many of these are surveyed in a recent book by Fish (2009).

Most relevant to the current thesis, there are a number of methods designed for phenomena where microscopic features are present throughout the domain and exhibit strong interactions with the macroscopic dynamics, rendering the latter very expensive to compute. The primary goal of such methods is to enable efficient and accurate computation of the large-scale behavior. This is performed by carrying out numerical simulations of the fine-scale features on much smaller test domains, in order to estimate their effect on the large-scale dynamics; these techniques are known as *averaging* and *homogenization*. Such a procedure is only possible in systems where there is a scale separation between the macroscopic and microscopic features. For example, simulation problems with rapidly varying coefficients, such as elasticity of composite materials (Milton, 2002), complex polymeric fluids (Ren and E, 2005), and porous media flow (Kippe et al., 2008), can be solved through such an approach. For further discussion of these methods, the reader is referred to the excellent reviews by Pavliotis and Stuart (2008) and E et al. (2007).

Multiscale methods as described above form a powerful numerical framework for accurate macroscopic simulation of multiscale phenomena. However, these techniques may not provide a direct benefit for many visual applications. Firstly, their computational advantage relies on restricting the fine-scale model to regions much smaller than the entire simulation domain. Secondly, they rely on a significant separation of scales between the microscopic features and the large-scale macroscopic dynamics of interest. In visual applications such as animation, both these properties are disadvantageous. As realistic animation requires modeling fine detail on the entire simulation domain, multiscale methods cannot offer a computational advantage over direct numerical simulation at a fine resolution. Further, many of the phenomena of interest here, such as turbulent fluids and crowds, do not show a wide separation between coarse and fine features, but rather a continuum of features with complicated dynamics at all scales. Therefore, while the *philosophy* of multiscale modeling is closely related to the problem of *visual* simulation of multiscale phenomena, the latter problem requires novel and distinct techniques for an

efficient solution.

1.3 THESIS STATEMENT

My thesis is as follows:

> *Visual simulation of complex multiscale phenomena such as turbulent fluids, granular materials, and human crowds can be performed efficiently and practically on current commodity hardware through scale decomposition techniques that couple together separate mathematical models for the global motion and fine-scale detail of these dynamical systems.*

The approach of scale decomposition introduced in this thesis rests on two key ideas: the use of distinct mathematical models for different scales of a single phenomenon, and the coupling between these models to take into account the interaction between scales.

In multiscale phenomena, the wide variety of qualitatively different behavior at different scales proves to be a central difficulty for existing modeling and simulation techniques. However, if we decompose the complex features spanning this wide range of scales into smaller subranges, for example, into a range of macroscopic or large-scale features, and a range of microscopic or fine-scale features, this computational challenge can be transformed into simpler subproblems. This approach provides the freedom to choose appropriate and efficient models that are closely tuned to the behaviors desired to be captured at each scale.

For example, consider the problem of modeling the flow of an energetic, turbulent fluid that includes features several meters in size as well as centimeter-scale details. One may choose to represent the large-scale dynamics of the fluid using a coarse physically-based numerical solver, which for efficiency only resolves features the size of a tenth of a meter. This may be sufficient to captures the visually prominent physical interactions of the fluid with boundaries, other objects in the environment, and other external forces.

Simultaneously, a simpler technique can be applied to model the chaotic fine-scale features such as turbulent vortices, ripples, and splashes. A large amount of prior work in graphics has shown that these can be convincingly modeled using extremely fast procedural techniques. Thus, an approach that combines these methods makes it practical to simulate highly detailed fluids. In particular, if the resolutions of the large-scale solver and the fine-scale model are m and n elements in each linear dimension respectively, with $m \ll n$, the cost of this combined approach for fluid simulation is only $\Omega(m^4) + O(n^3)$, as compared to $\Omega(n^4)$ for a comparable resolution of numerical simulation alone. Similar computational benefits are seen for other multiscale phenomena modeled using this approach.

The second essential component of this thesis is the coupling between these two models, which are typically based on very different mathematical representations. For example, one may wish to combine data-driven and procedural models with physically-based numerical simulation, or discrete representations of microscopic entities with continuum-based models of macroscopic flow. In each case, it is necessary to connect the different models at different scales through some form of coupling, so that both models form a consistent representation of the system as a whole, and so that the interactions between features at different scales is taken into account. The formulation of the coupling technique for any specific application depends on the specifics of the models used at different scales, as well as on the nature of the interactions across scales in the actual multiscale phenomenon to be modeled. I will describe coupling techniques for four different applications covering a broad range of different combinations of macroscopic and microscopic models.

1.4 MAIN RESULTS

The thesis presented above describes a broad conceptual approach for modeling and simulation of multiscale phenomena for visual applications. In this dissertation, I show four specific applications of this approach to a range of challenging, multiscale simulation

problems. Two techniques are presented for highly detailed animation of fluids showing fine-scale surface features and complex turbulent dynamics. For simulating the motion of large aggregates of discrete entities, such as dense crowds of pedestrians, and granular materials such as sand and powders, I describe two closely related techniques. In all of these applications, the proposed approach yields a significant performance gain over traditional techniques for numerical simulation.

1.4.1 TEXTURE SYNTHESIS ON FLUIDS

Real-world fluids exhibit complex surface behaviors, such as small ripples in a stream, foam and bubbles in turbulent flow, and crust patterns in lava. These features cannot be easily reproduced by traditional simulation techniques without significant modeling effort and a high computational cost.

I describe a novel approach that combines numerical fluid simulation with texture synthesis to capture such detailed fluids for realistic animation. Here, the physical motion of the fluid is computed through direct numerical simulation. The visual appearance of fine-scale surface features is modeled as a dynamic texture over the fluid's surface, generated through optimization-based texture synthesis using footage of real fluids. Novel techniques are introduced to couple the texture synthesis process to the simulated large-scale flow. With this approach, realistic fluid detail can be rendered using textures that vary over space and time in correlation with the physical characteristics of the fluid motion, and also exhibit their own intrinsic dynamics derived from the fine-scale dynamics of the input footage.

This is the first technique which generates complex physically-based flow appearance through texture synthesis. It combines physical and geometric features, controllable texture synthesis on continuous flows, and animated texture synthesis using video input, using a single framework for spacetime texture optimization. This is a very general approach, capable of producing visually convincing results for diverse physical and visual

effects with little user intervention.

1.4.2 TURBULENCE MODELING AND ANIMATION

Obtaining true three-dimensional detail in the motion and geometry of the simulated fluid is not possible through texture alone. Numerical simulation for such fine detail requires a mesh with very high spatial resolution, rendering the associated computational cost prohibitively high. As an alternative, procedural techniques exist for animating turbulent fluids, but they require careful animator control and are not applicable to completely general flow scenarios as they cannot reproduce physical interactions between fluids and the environment.

I present a novel method that combines numerical fluid simulation for the macroscopic flow with a procedural model for fine-scale turbulent eddies to achieve fast, highly detailed visual simulation of fluids. The procedural turbulence model is faithful to known statistical properties of real turbulence such as the Kolmogorov spectrum of kinetic energy. To couple the numerical and procedural representations, I introduce a novel energy model which explicitly tracks the transfer of kinetic energy from large to small scales. This model, derived from existing theories in the fluid dynamics literature, captures the qualitative features of turbulence as it evolves due to the motion of the fluid. Another contribution is a method for backwards coupling, which allows the fine-scale turbulence to introduce fluctuating eddies and vortices into the numerically simulated flow through an overlapping range of scales.

This approach both compensates for the loss of detail due to numerical dissipation and introduces additional subgrid-scale flow which cannot be captured on the simulation grid at all. It accounts for the two-way interaction of the procedurally synthesized turbulence with the coarse simulated flow, enabling the turbulent flow to affect the dynamics of the numerical simulation for added realism. The technique is general and applicable to a wide spectrum of fluids, including swirling gaseous flows and complex river rapids with

free surfaces. Finally, it yields fine-scale fluid details from procedural synthesis while requiring only a low-cost fluid solver on a coarse mesh, achieving an order of magnitude performance gain over high-resolution fluid simulation that produces comparable details.

1.4.3 SIMULATION OF DENSE CROWDS

Human crowds exhibit highly complex behavior driven by individual decisions of agents with respect to their goals, environmental obstacles, and other nearby agents. I focus specifically on the problem of simulating the inter-agent dynamics of large, dense crowds in real time. One major subproblem for crowd simulation in general is that of local collision avoidance: determining the velocities of all agents that prevent collisions with each other. Existing techniques for local collision avoidance require each agent to take into account the motion of its nearby neighbors; this step quickly turns into a major computational bottleneck for very dense crowds.

It has been observed in the literature that dense crowds exhibit a low interpersonal distance and a corresponding loss of individual freedom of motion. Based on this observation, I propose a technique that models the behavior of such crowds efficiently as a continuous flow on the macroscopic scale, coupled with a traditional fine-scale representation of discrete individuals. These two models are explicitly coupled by transferring density and velocity information directly between them. The computational advantage of this approach comes from a novel mathematical formulation, termed unilateral incompressibility, which transfers local collision avoidance into the continuum domain, decoupling its computational cost from the number of agents.

This approach enables simulation of very large crowds consisting of hundreds of thousands of agents at interactive rates. When introduced, it was an order of magnitude faster than existing state-of-the-art techniques for crowd simulation, and showed unprecedented performance for extremely large crowds of up to a million agents. This work also introduced the unilateral incompressibility formulation, which is a very general model for

representing the collective flow of large aggregates such as hair, piles of objects, dense traffic, and granular materials.

1.4.4 CONTINUUM SIMULATION OF GRANULAR MATERIALS

Granular materials such as sand, powders, cereal grains, and gravel are commonplace in the physical world around us. Being composed of very large numbers of rigid grains, they show unique physical behavior that is unlike other materials such as fluids and deformable bodies, which have been well studied in computer graphics. In particular, granular materials disperse freely in free fall, flow plastically under forcing, and yet settle in stable piles.

The physical behavior of such materials arises from the interplay of contact and frictional forces between millions of tiny grains. Simulating the motion of each such grain is computationally prohibitive for large scenarios or fine-grained materials like sand. I propose an approach based on a continuum model, which treats a granular material as a continuous fluid. This is combined with a point cloud of individual grains for rendering, and an intermediate particle-based model that couples these two representations. In order to capture the unique behaviors that such materials exhibit, I present novel contributions that depart significantly from traditional fluid simulation techniques. Firstly, to allow the material to disperse freely when agitated but maintain its volume when at rest, the assumption of incompressibility used for fluids is replaced with a unilateral variational constraint similar to that used above for simulating crowds. Secondly, unlike fluid viscosity, friction in granular materials can counteract gravity to maintain stable piles in equilibrium. Modeling this behavior requires solving for the internal stresses in a global fashion, for which I present an efficient numerical technique.

These contributions permit much more realistic and efficient simulation of granular flow than previously possible. It is as fast as previous continuum-based techniques, but reproduces a wider range of granular behaviors, including freely dispersing flow without

artificial cohesion, a robust handling of friction, and two-way interaction with solid objects. These visually prominent features could previously only have been modeled using computationally expensive discrete models, computing the interactions between individual grains; the presented technique is an order of magnitude faster than such methods.

1.5 ORGANIZATION

The remainder of this dissertation is organized as follows. Chapters 2 to 5 deal, in turn, with each of the techniques introduced in the previous section. Each chapter describes existing methods for the relevant problem, the large-scale and fine-scale models used in the proposed technique, as well as the coupling between them. Chapter 6 summarizes the main contributions of this work, and discusses limitations and open directions for future research.

CHAPTER 2

TEXTURE SYNTHESIS ON FLUIDS

2.1 INTRODUCTION

The realistic simulation and rendering of physical fluids such as liquids, viscous flows, flames, and smoke has been an important but challenging problem in computer graphics. Real fluids exhibit many complex, detailed behaviors, both in their surface appearance and in the motion throughout the fluid. In this chapter, I will focus on the visual appearance of surface features on liquids. Examples include small ripples in a stream, foam and bubbles in turbulent flow, and crust patterns in lava, which cannot be easily reproduced by traditional simulation techniques. The visual appearance of these phenomena can be thought of as dynamic textures over the flow surfaces, similar to the traditional graphics technique of rendering surface detail on static objects using texture mapping. On fluids, however, these textures move with the flow on the dynamically changing fluid surface, and vary over space and time in correlation with the underlying physical characteristics of the flow.

This chapter describes a technique for combining numerical simulation of fluids with texture synthesis to render such fine-scale surface effects. The physical and geometric characteristics of the flow are used to guide the synthesis of dynamic, non-uniform textures over its domain. These characteristics are derived from the surface properties and velocity fields of the simulated liquid, and are used to model the behavior of surface flow

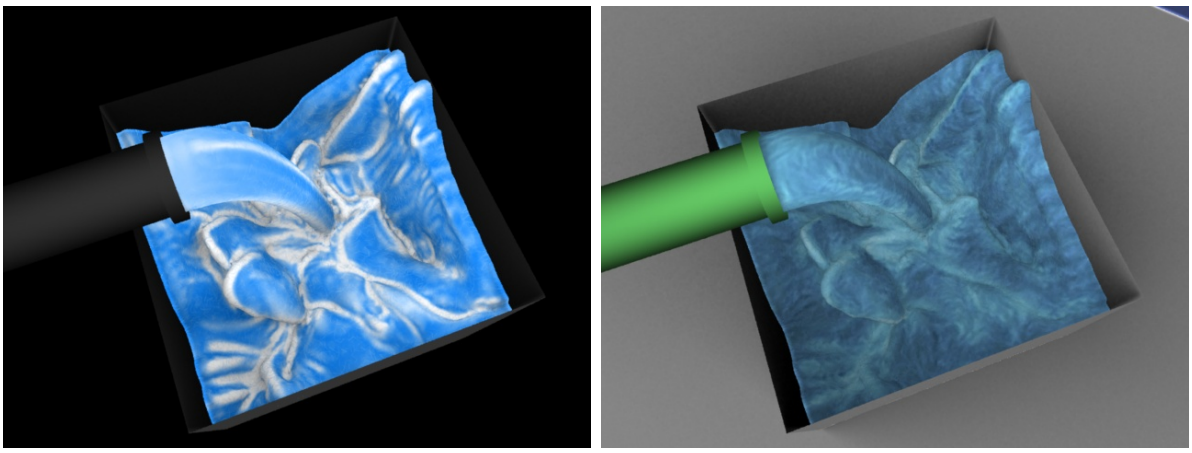


Figure 2.1: Water pours from a hose into a box. A dynamic, spatially varying texture reproduces small-scale phenomena such as foam and ripples on the liquid surface. The feature map shown in (a) is computed using physical characteristics of the flow, and is used to guide texture synthesis to generate visual effects as shown in (b).

phenomena over space and time. This model then automatically controls the synthesis of a spatially and temporally varying texture over each frame of the simulation or video clip using multiple, dissimilar input textures, as shown in figure 2.1.

This work, first published as Narain, Kwatra, Lee, Kim, Carlson, and Lin (2007), was the first technique which generates complex physically-based flow appearance through texture synthesis. It combines physical and geometric features, controllable texture synthesis on continuous flows, and video textures, using a single optimization-based texture synthesis framework. With this approach, evolving, heterogeneous textures over arbitrary flow surfaces can be generated in a physically consistent and visually convincing manner. This framework for feature-guided texture synthesis is a very general approach, capable of producing visually convincing results for diverse physical and visual effects with little user guidance.

The rest of this chapter is organized as follows: In section 2.2, I summarize the related work in relevant areas. Section 2.3 presents the technique of feature-guided texture synthesis for generating heterogeneous textures controlled by the fluid behavior. In section 2.4, I describe an extension of this approach to use real video footage to capture

texture evolution, in order to perform synthesis of animated textures. Finally, I present the results of this technique for a variety of different scenarios and effects in section 2.5.

2.2 RELATED WORK

Texture synthesis has been extensively studied in the context of novel image and video creation and manipulation. A pertinent extension of traditional texture synthesis methods is the use of flow fields to guide the motion of texture elements for synthesizing animated textures. This capability was demonstrated by Stam (1999) who advected solid textures through a fluid flow field, and Neyret (2003), who pioneered the self-regenerating advection of unstructured textures over flow fields in 2D and 3D. Newer approaches are able to perform this on more complex textures in the 2D domain (Bhat et al., 2004; Kwatra et al., 2005). Recently, a few authors have proposed techniques for advection of more general textures on animated 3D surfaces (Kwatra et al., 2007; Bargteil et al., 2006b; Han et al., 2006). These works focus on generating homogeneous textures that do not necessarily exhibit meaningful variation over space and time.

Many texture synthesis methods in 2D (Efros and Freeman, 2001; Ashikhmin, 2001; Kwatra et al., 2003, 2005) and 3D (Zhang et al., 2003) are also capable of performing controllable texture synthesis according to user-specified constraints. The use of an explicitly specified feature map for controlling texture synthesis was introduced in the context of texture transfer (Hertzmann et al., 2001; Efros and Freeman, 2001). The idea of generating a feature map automatically using geometric characteristics of static meshes was proposed by Mertens et al. (2006). Salient features can also be computed on the input textures, which can improve synthesis quality (Wu and Yu, 2004) and allow blending between textures (Zhang et al., 2003).

Some related methods for video manipulation treat a full frame as a unit (Schödl et al., 2000; Doretto and Soatto, 2003). Other works (Wei and Levoy, 2000; Kwatra et al., 2003) treat video texture synthesis as a 3D texturing problem, producing a spatio-temporal

volume. Bhat et al. (2004) track the evolution of texture patches along stationary flow lines in the input and render them with the desired motion into the output video.

An alternative to the use of textures for reproducing surface detail is to add techniques to the simulator itself to explicitly model the desired effects. Several such methods have been proposed for effects such as bubbles (Hong and Kim, 2005; Greenwood and House, 2004) and foam (Takahashi et al., 2003; Kim et al., 2007a). However, these methods often require deep modifications to the fluid simulation code. They are also less versatile than textures, and cannot be easily adapted to other fluid features, such as the stretching and cracking of lava, or effects such as non-photorealistic texturing which are not physically based, in a single framework.

2.3 FEATURE-GUIDED SYNTHESIS

This work described in this chapter builds on existing optimization-based frameworks for texturing of surfaces and fluids (Kwatra et al., 2007; Bargteil et al., 2006b; Han et al., 2006), which synthesize high-quality textures on fluid surfaces from input texture exemplars. In contrast to these approaches, the method I present extracts salient features using physical characteristics and geometric properties of the flow to model the temporally and spatially varying patterns of complex natural phenomena.

The basic concept of this approach is based on the observation that texture elements on a flow surface, such as ripples and foam on turbulent water, show variation on both large and small scales of space and time. Traditional texture synthesis only takes into account the small-scale spatial variation of the texture. The large-scale variation of texture on real flows is typically correlated with the physical characteristics of the flow. On fluids, texture also displays small-scale evolution over time which cannot be captured by using static images as texture exemplars. I propose a general framework which integrates flow features and video exemplars seamlessly in the optimization-based texture synthesis technique. Here, a time-varying video texture is treated as a Markov process whose future

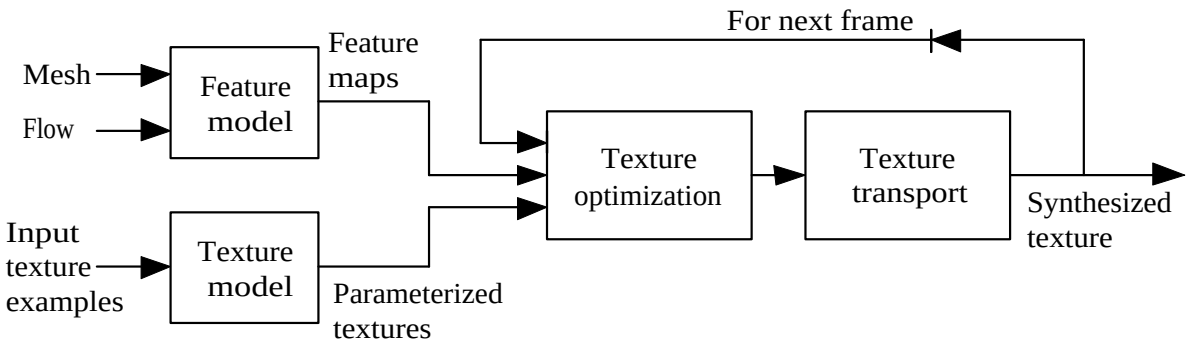


Figure 2.2: An overview of the technique for feature-guided texture synthesis on flows.

state can be determined by its current and past states.

The main elements of the system are shown in figure 2.2. The input is a sequence of animated fluid surface meshes and the time-varying velocity field of the flow. For each frame, the physical features of the fluid are used to compute a *parameter map* over the surface, which defines the spatial variation of texture appearance. This parameter map is used to guide the texture synthesis process on the surface based on the texture and video input. The texture and the parameter map are then transported via the flow to the corresponding locations on the next frame to be used for maintaining temporal coherence.

2.3.1 SYNTHESIS ON FLUIDS

I briefly review the existing approach for synthesis on fluids using homogeneous textures as input exemplars. For a flow sequence, texture synthesis is performed one frame at a time, treating the flow surface at a particular frame as a static mesh. As a preliminary step, the surface is covered uniformly with overlapping patches to facilitate comparison with the 2D input texture. The output texture is then generated by minimizing an energy function $E(\mathbf{s};\mathbf{p})$ which measures the similarity of the texture at each surface patch $\mathbf{s}$ with a similar patch $\mathbf{p}$ in the input exemplar.

For temporal coherence, the generated texture from the previous frame is advected using the known velocity field over time to yield a target texture, which is used as a soft

constraint in the texture optimization step. Thus the net energy function for a surface patch $\mathbf{s}$ is

$$E(\mathbf{s}) = E_{\text{texture}}(\mathbf{s}; \mathbf{p}) + E_{\text{coherence}}(\mathbf{s}; \mathbf{s}') \tag{2.1}$$

where $\mathbf{s}'$ is the advected texture color. These energy functions measure the intensity/color discrepancy between the surface patch and the corresponding input and advected patches respectively, and generally take the following form:

$$E_{\text{texture}}(\mathbf{s}; \mathbf{p}) = ||\mathbf{I}(\mathbf{s}) - \mathbf{I}(\mathbf{p})||^2 \tag{2.2}$$

$$E_{\text{coherence}}(\mathbf{s}; \mathbf{s}') = ||\mathbf{I}(\mathbf{s}) - \mathbf{I}(\mathbf{s}')||^2 \tag{2.3}$$

where $\mathbf{I}$ refers to the intensities or colors associated with a patch. The total energy over all patches, $\sum_{\mathbf{s}} E(\mathbf{s})$, can be minimized by an iterative approach. For a given (initially random) assignment of input patches to surface patches, the surface texture can be readily obtained by minimizing E independently at all points on the surface. Then, the assignment of input patches is updated by finding, for each surface patch, the patch in the input texture that minimizes E_{texture}. These steps are repeated until the pach assignment no longer changes. Further details are given in previous work by Kwatra et al. (2005, 2007) and Bargteil et al. (2006b). Additionally, techniques such as multi-resolution synthesis (Wei and Levoy, 2000) and appearance-space transformation (Lefebvre and Hoppe, 2006) can be used to improve synthesis quality within this framework.

The process of constructing local patches on the surface for texture synthesis requires a smooth local orientation field at each point on the surface. This field also controls the orientation of the output texture. To maintain temporal coherence of texture orientation, the orientation field is advected across frames using a modified advection procedure (Kwatra et al., 2007).

The texture synthesis technique described above is only capable of generating textures which are roughly homogeneous over space and time. While two or more textures can be used, as in Bargteil et al. (2006b) for example, this approach is limited to assignment

at the initial state, beyond which they cannot be varied dynamically. The result is that while the synthesized textures show temporal coherence and correctly move with the fluid flow, they are too uniform over space and time to convincingly display dynamically evolving textures. Below, I present a novel contribution for synthesis of spatially varying textures.

2.3.2 FLOW FEATURES

The spatial and temporal variation of texture has to be controlled using the physical features of the flow itself. This is performed through the use of certain metrics to characterize such properties of the flow, which are termed *features*. The values of these features over the flow surface can then be used to guide the texture synthesis process. The choice of metric typically depends on the type of flow and the specific effect to be generated.

Basic Principles: There are several possible metrics that can be used for controlling the texture synthesis. One can use arbitrary combinations of such metrics to allow the various kinds of effects to be simulated. Experimentation reveals, however, that better results are obtained from features which are physically motivated by the physics of the desired effects rather than from *ad hoc* metrics. Below, I discuss specific examples of such features which were found generally useful for applications of fluid texture synthesis.

Curvature: The mean surface curvature of the fluid surface is commonly used as a measure for adding splashes and foam to water simulations (Fournier and Reeves, 1986; Kim et al., 2006). Therefore, this is a natural metric use for modeling foam in the texture synthesis approach. This feature is described further in section 2.3.4.

Divergence: For simulating viscous fluids such as lava flow, the visually significant behavior is the anisotropic stretch and squash of the fluid surface. This behavior causes

effects such as cracking and crumpling of the lava crust. One metric which can be used in this case is the divergence of the velocity flow. Since only the surface behavior of the flow is of interest, the divergence operator should be projected onto the tangent plane at the surface, so that

$$\nabla_t \cdot \mathbf{u} = \partial_1(\mathbf{e}_1 \cdot \mathbf{u}) + \partial_2(\mathbf{e}_2 \cdot \mathbf{u}) \tag{2.4}$$

where ∂_1 and ∂_2 denote the directional derivative along orthogonal tangent vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ at the surface. This is also desirable since for an incompressible flow, $\nabla \cdot \mathbf{u}$ is identically zero while $\nabla_t \cdot \mathbf{u}$ may not be.

Jacobian: The Jacobian matrix of the velocity field is another useful characteristic which is used in, for example, flow visualization (Post et al., 2003). Again, only the tangential component $\mathbf{J}_t$ is considered:

$$\mathbf{J}_t = [\partial_j(\mathbf{e}_i \cdot \mathbf{u})]_{2\times 2}. \tag{2.5}$$

$\mathbf{J}_t$ completely characterizes the local spatial variation of the velocity in the tangent plane. In particular, the eigenvalues of its symmetric component, defined as

$$\mathbf{J}_t^+ = \frac{1}{2}(\mathbf{J}_t + \mathbf{J}_t^T), \tag{2.6}$$

measure the extremal values of directional divergence. These eigenvalues can be a more informative measure of flow behavior than the divergence alone, especially for viscous flow, such as lava, where the directional stretching and compression of the fluid is relevant.

Internal forces: The fluid experiences internal stresses due to viscous effects and pressure gradients, which can be estimated from the flow field. For example, the viscous force is given by $\mathbf{F}_{\text{viscous}} = \eta\nabla^2\mathbf{u}$.

These are not the only useful features for controllable texture synthesis. Depending on the specific application, any other metric which can be evaluated on the fluid surface can be used. For example, for simulating lava, the surface temperature is an important

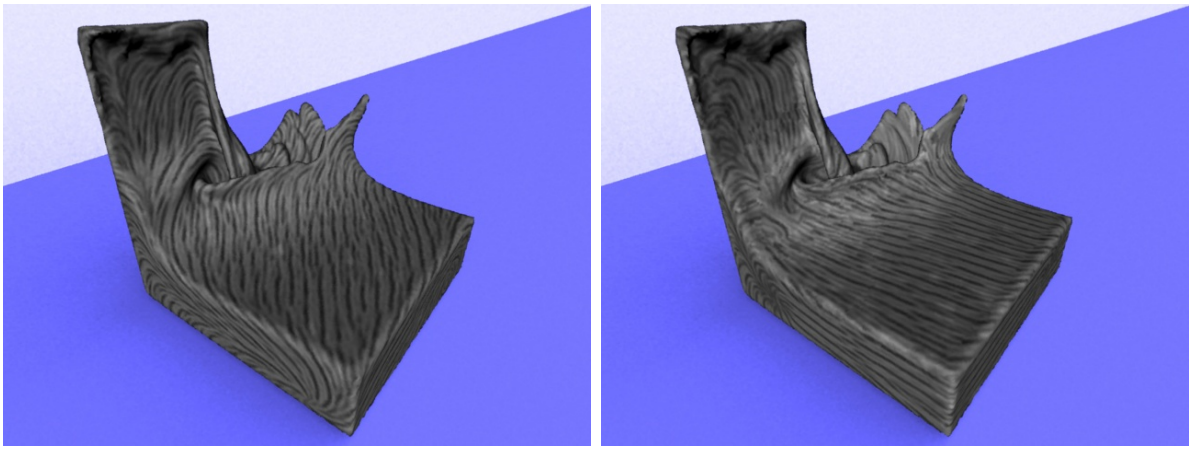


Figure 2.3: Flow-based control of orientation. (a) Without orientation control, the orientations are arbitrary and do not reflect the flow characteristics. (b) Using flow features, a coherent orientation field is obtained which is correlated with the flow.

feature which influences the appearance of the fluid. The user is free to include this as an additional feature, whether approximated as a function of position or computed as part of the underlying simulation.

2.3.3 ORIENTATION CONTROL

It is desirable in the case of strongly directional flow to determine the orientation of the advected texture accordingly. This control becomes important when synthesizing anisotropic textures, since the direction of anisotropy is typically related to the characteristics of the flow. Examples include the patterns formed on the crust of flowing lava, which are caused by non-uniform stretching or compression of the surface in different directions.

In order to orient the texture patches along the appropriate directions, one requires a field of tangential directions over the fluid surface. I propose that for texturing flows, the most important directions are extrema of directional divergence, since under the influence of the flow, a small linear element will tend to align over time with the direction of maximum divergence. These extrema are directly given by the eigenvectors of the tangential Jacobian matrix $\mathbf{J}_t$.

In practice, it is preferable to use the symmetric component $\mathbf{J}_t^+$ rather than the full tangential matrix $\mathbf{J}_t$, which ensures that the eigenvectors are real everywhere and orthogonal to each other. Since these eigenvectors are unstable at regions where the eigenvalues are small and/or nearly equal, one should not directly set the orientation field $\mathbf{d}$ at each point to be equal to the maximal eigenvector, say $\mathbf{v}$. Instead, one may apply a force to adjust its direction towards $\mathbf{v}(\mathbf{x})$ over time:

$$\frac{\partial \mathbf{d}}{\partial t} = \mathbf{D}(\mathbf{d}, \mathbf{u}) + \lambda(\mathbf{I} - \mathbf{d}\mathbf{d}^T)\mathbf{v} \tag{2.7}$$

where $\mathbf{D}$ is the modified advection operator for transport of orientation fields (Kwatra et al., 2007). The strength λ of the force is set to be proportional to the anisotropy of the flow, which can be measured through the stability of $\mathbf{v}$ relative to perturbations of $\mathbf{J}_t$. This adjustment ensures that the orientation field is governed by the maximal eigenvector only in regions where the flow is strongly anisotropic.

2.3.4 FEATURE-GUIDED TEXTURES

Once the flow features used to vary the synthesized texture over the surface have been determined, they have to be integrated into the texture optimization process in order to generate non-uniform textures based on these features. Thus, it is necessary to relate the fluid features, which are continuous in nature, to a finite set of input textures.

Let us adopt the formulation used in existing work on texture transfer (Hertzmann et al., 2001; Efros and Freeman, 2001; Mertens et al., 2006). Given a heterogeneous set of texture exemplars, each input patch is associated with a point $\mathbf{r}$ in a low-dimensional parameter space which characterizes the range of variation of the set of texture exemplars. For example, a spatially varying foam texture can be parametrized by the density of foam, while a texture representing lava could vary according to surface temperature and the solidity of the surface crust. In other words, one views the input textures as a set of samples of a low-dimensional family of textures.

To determine the spatial variation of the generated texture, corresponding parameter values must be defined over the surface, which the input textures can be matched against. I call this set of parameter values the *parameter map* (similar to the "correspondence map" of Efros and Freeman (2001) and the "guidance field" of Mertens et al. (2006)). Using this parameter map to guide texture synthesis is then straightforward: an additional energy function is introduced,

$$E_{feature}(\mathbf{s};\mathbf{p}) = ||\mathbf{r}(\mathbf{s}) - \mathbf{r}(\mathbf{p})||^2, \tag{2.8}$$

which measures the similarity of feature values on the surface to those of the selected input patches. This acts as a soft constraint, favoring the selection of texture patches which have similar parameter values to those on each surface patch. One should not enforce that the parameters are exactly matched, or that the patch with the nearest parameter values is chosen, because the parameter values associated with the input textures are usually sparse and a nearest match may not be meaningful. Using a soft constraint allows parameter matching to be traded off against texture smoothness across changes in the parameter map.

My contribution in this part lies in relating this parameter map to the feature values obtained from the physical behavior of the fluid. Note that there is a semantic gap between the feature values, say $\mathbf{q}$ in vectorized form, which are typically physical quantities, and texture parameters $\mathbf{r}$, which are more qualitative in nature. The relation between them depends on the application. For example, for turbulent water, relevant feature values include surface curvature, while texture parameters include the density of foam on the surface, with the desired relationship being that high curvature values (such as splashes and breaking waves) should lead to the increase of foam density.

While it is possible to set $\mathbf{r}$ directly as a function of the instantaneous feature values $\mathbf{q}$, I found that this does not yield convincing results as it lacks the proper temporal behavior: for example, foam would disappear immediately when the curvature decreased. Therefore, it is desirable that the feature values instead govern the way the parameters

change over time. This can be realized by defining a first-order differential equation of the form

$$\frac{D\mathbf{r}}{Dt} = f(\mathbf{q}, \mathbf{r}).$$

Here the Lagrangian derivative is used so that the parameter map is also transported with the flow in a realistic manner. In practice, this can be implemented by first advecting the parameter map via the flow field, and then updating its value using $f(\mathbf{q}, \mathbf{r})$.

As described in the previous sections, the texture properties relevant to the flow features may need to be determined on a case by case basis. While this aspect may be considered as a limitation if the goal is to achieve complete automation, it also empowers the animator to make artistic choices while using the system and therefore enhances its usability. Note that the texture/feature relationship needs to be determined only once for a given texture, and can be quite simple for static textures, such as the assignment of a discrete label such as "turbulent", "calm", "intermediate" and so on.

2.4 VIDEO TEXTURES

Many fluid phenomena exhibit meso-scale evolution over time, which is distinct from passive advection. For example, foam and ripples on a water surface are not simply fixed to only move with the water's large-scale flow, but also change over time in a characteristic manner. To faithfully reproduce such behavior, I propose the use of videos of real fluids as exemplars for the texture synthesis process. In real videos, the fluid has its own flow which also transports the texture over time. I describe a method to factor out this flow to obtain stationary input exemplars, and a novel video texture model to generate new animated textures which show similar temporal behavior.

2.4.1 MOTION REMOVAL

To decouple meso-scale texture evolution from the fluid flow in the video, it is necessary to determine the flow field of the input video using a motion estimation technique. There

are two main methods for motion estimation, namely optical flow (Schunck and Horn, 1981; Black and Anandan, 1996) and feature tracking (Tomasi and Kanade, 1991; Shi and Tomasi, 1994; Lowe, 2004). Both approaches were tested on sample videos of fluids, and optical flow was found to work consistently better for these examples. This may be due to the nature of fluid videos, which have non-rigid motions and do not show many sharp feature points that can be tracked consistently.

The output from the motion estimation procedure yields a dense flow field which accurately captures the temporal changes in the video. However, since the motion estimation cannot distinguish between large-scale advection and texture evolution, both of these are reflected in the output motion field. Therefore, to remove the effect of the latter, a Gaussian smoothing step is performed on the motion field to retain only the large-scale motion of the fluid in the video as the flow estimate.

2.4.2 SYNTHESIS OF EVOLVING TEXTURES

For video textures to be used to model fine dynamical features on liquids, there are two requirements: the technique should be able to generate heterogeneous spatially and temporally varying textures, and do so in a manner that fits naturally with the existing patch-based surface texture synthesis algorithm. Existing techniques for video texture synthesis are not easily extended to this domain. The technique I present below allows for video textures to be supported in the texture optimization framework, thus enabling such animated textures to be synthesized with high texture quality.

In order to synthesize an evolving texture whose behavior matches that of the input video, one requires a model of how the input video changes over time, so that given the current state of the texture, its likely appearance in the next frame can be predicted. A stochastic video texture can be modeled as a Markov process, in that its appearance at a given frame is determined to a large extent by the appearance at the previous frame. This assumption holds true of textures generated by physical processes, if one allows for

hidden state variables which may not be directly reflected in the texture colors. However, a hidden variables model is difficult to infer, as only color values are known from the input.

Instead, a higher-order Markov model may be used, where the appearance of a texture patch at one frame is determined by its appearance at not just one but n previous states. Such a model can be built implicitly by using tuples of $n+1$ corresponding patches $[\mathbf{p}_n, \ldots, \mathbf{p}_1, \mathbf{p}_0]$ from consecutive frames of the video as samples of its temporal behavior. I call such tuples *video patches*, since they form the basic unit of representation for video textures, analogous to 2D texture patches for static textures. The set of video patches is constructed by selecting corresponding image patches on every set of $n+1$ consecutive frames, after tracking and warping to compensate for the estimated flow field. The number of previous frames n used determines the degree of continuity in the synthesized texture, and a small value of up to 3 should be sufficient for most applications. A similar model was used in previous work by Schödl et al. (2000) to correctly account for dynamical continuity.

For handling texture evolution on a surface, the texture state is represented by storing, at each vertex, not just the current color but the history of the current and past color values over n frames. For synthesizing a new frame, recall that due to the transport procedure the texture colors from the previous frames, say $\mathbf{s}_{n+1}, \ldots, \mathbf{s}_1$, are known. Since the evolution of a surface patch should closely match that of the input video patches, the energy function for the current texture of the surface patch $\mathbf{s}_0$ with respect to a selected video patch $\mathbf{p} = [\mathbf{p}_n, \ldots, \mathbf{p}_0]$ is defined simply as

$$E(\mathbf{s}; \mathbf{p}) = \sum_{i=0}^{n} w_i^2 (||\mathbf{I}(\mathbf{s}_i) - \mathbf{I}(\mathbf{p}_i)||^2 + ||\mathbf{r}(\mathbf{s}_i) - \mathbf{r}(\mathbf{p}_i)||^2) \tag{2.9}$$

where $w_n, \ldots, w_0$ are weights that can be used to control the relative importance of previous frames in the energy function. Texture synthesis over the entire surface is performed by minimizing the sum of this energy function over all patches, varying only the current texture $\mathbf{s}_0$.

Since this new energy function is also quadratic, it can still be solved using the same fast least-squares solvers used in existing work. Specifically, the texture synthesis approach still has the same two steps as before. In each iteration, first an input patch $\mathbf{p}$ is found for each surface patch $\mathbf{s}$ which most closely matches the previous (and current, if known) values; then $\mathbf{s}_0$ is determined for all surface patches to minimize the total energy $\sum_{\mathbf{s}} E(\mathbf{s}; \mathbf{p})$.

This approach combines both the main texture similarity energy and the texture evolution constraint into a single consistent formulation, and automatically accounts for temporal coherence as well. For a given surface patch, the selected input video patch $\mathbf{p}$ chosen will be one which matches the previous patches $\mathbf{s}_n, \ldots, \mathbf{s}_1$. If the input patches are temporally coherent so that $\mathbf{p}_0$ is similar to $\mathbf{p}_1$, then $\mathbf{s}_0$ necessarily must be correspondingly close to the previous frame's $\mathbf{s}_1$. On the other hand, if the input video contains discontinuities between frames, then the texture will also behave similarly. I believe this is a desirable behavior, and that the fact that temporal coherence as a whole arises naturally from this formulation without an additional term is a compelling benefit.

While this procedure has been developed for the synthesis of video textures on fluid surfaces, it is not restricted to this particular application. It can be used for the synthesis of any time-varying texture, whether in the 2D domain or on 3D surfaces. Also, the new energy function presented here can replace the original energy function even when only static image-based textures are used.

2.5 RESULTS AND DISCUSSION

This technique has been implemented in C++, and the results rendered using 3Delight®. It has been applied to several diverse scenarios with different fluids and input textures, demonstrating the variety of effects that can be produced by this technique. Below, a number of these examples are presented.

2.5.1 IMAGE TEXTURES

In figure 2.4, a large quantity of water is dropped on a broken dam, causing the water to splash. Three input textures are used with varying amounts of foam. The associated texture parameter on the surface, representing foam density, is controlled by the mean surface curvature H and decays gradually over time. An additional continuity term using $\nabla_t \mathbf{u}$ ensures that the total amount of foam is conserved under advection. Specifically,

$$\frac{Dr}{Dt} = k \cdot \max(\rho H - r, 0) - r/\tau - r\nabla_t \mathbf{u}$$

In this experiment, r was clamped to the range $[0, 1]$, and parameter values were set to ρ = 0.02 m, τ = 2 s, and $k \equiv \infty$ by directly assigning $r = \rho H$ if $r < \rho H$.

For comparison, the results of texture synthesis without taking fluid features into account are shown in figure 2.5. If features are not used, the texture is not correlated with the behavior of the fluid, and interestingly it converges to a local optimum of a uniform texture over time.

Figure 2.6 shows the different kinds of lava flow that can be generated by this technique. The first column shows an animation of hot molten lava cooling off over time, using a real image of lava with a user-designed feature map assigned. A spatially varying feature field which includes distance from the source is used to approximate temperature, and velocity divergence is used to model the clumping visible in the input image. The second column shows a colder lava flow with a solidified crust, using manually designed inputs textures. Here the maximum value of directional divergence is used to model the cracking of the crust.

2.5.2 VIDEO TEXTURES

Video textures are shown in figure 2.7. In this example, n was set to be 1; that is, only the immediately previous color was used in selecting the current color. The target features were determined in the same way as in figure 2.4. The feature map for the input

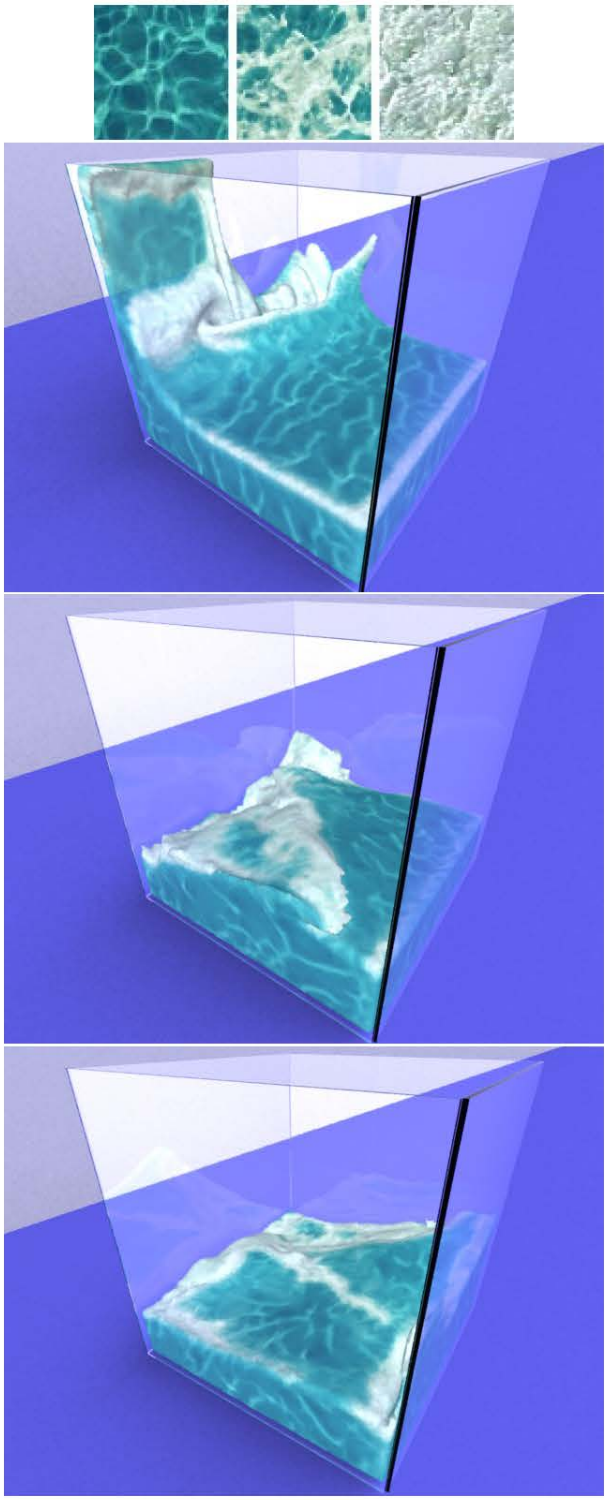


Figure 2.4: Texture inputs for water foam, and three frames of the synthesized foam texture on the broken dam simulation.

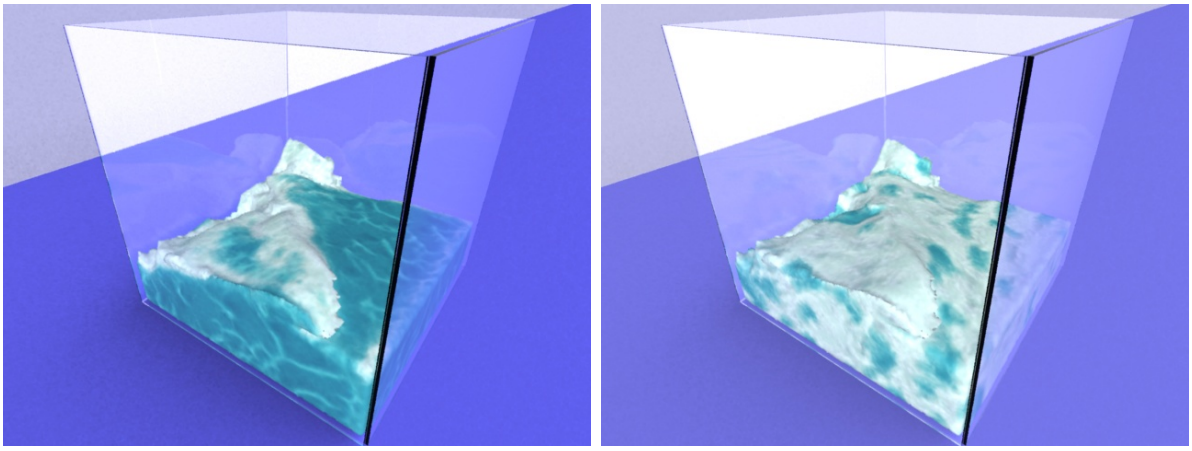


Figure 2.5: The broken dam simulation with feature guidance (a) enabled, and (b) disabled.

videos was specified manually. This technique can be used to add visual complexity to a simulation and yield an effective impression of turbulence.

Another application of this technique is to perform texture transfer between real videos for special effects. One input video can be used as the source of texture patches, and another target video to define the desired feature map and flow field. Performing texture synthesis restricted to the 2D grid of the target video results a video which has the appearance of one video and the temporal behavior of the other. Several results are presented in figure 2.8.

This technique is flexible enough to extend beyond synthesis of appearance only. Some preliminary experiments have been carried out with synthesizing complex, dynamic surface behavior during phenomena such as boiling. A boiling simulation is performed on a small $64 \times 64 \times 20$ grid representing a slice of volume near the surface of the fluid. This yields a sequence of distance fields representing the bubbling surface. This sequence is treated as a video texture with 20 channels—instead of the regular three (r, g, b)—where each channel stores the distance (from the liquid surface) at a location transverse to the slice. The 2D temperature field at the surface is treated as the feature map. Using "temperature maps" as input features to the system, one can synthesize the corresponding distance fields representing boiling fluids over much larger domains. In figure 2.9, I show

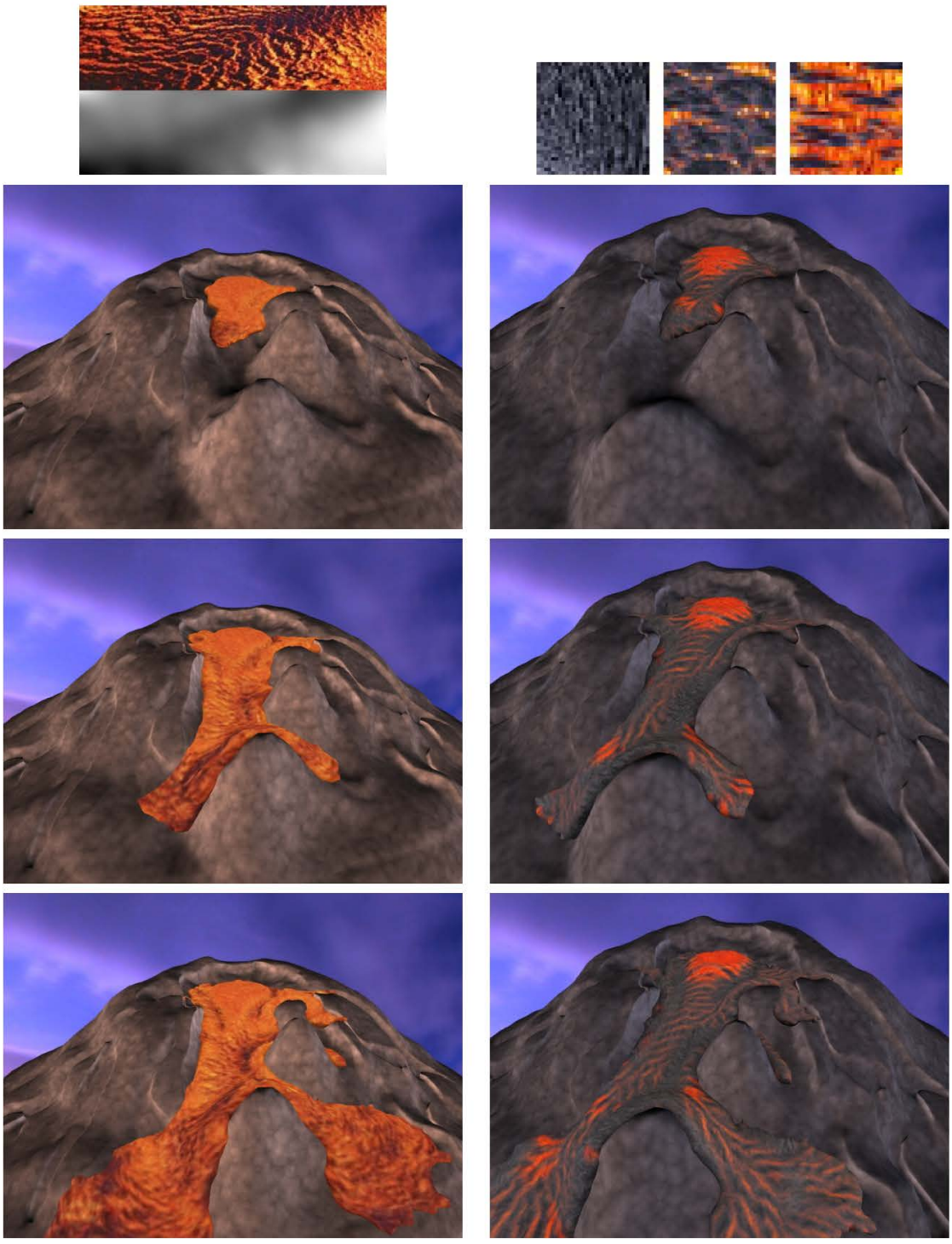


Figure 2.6: Two different kinds of lava flowing in complex environments.

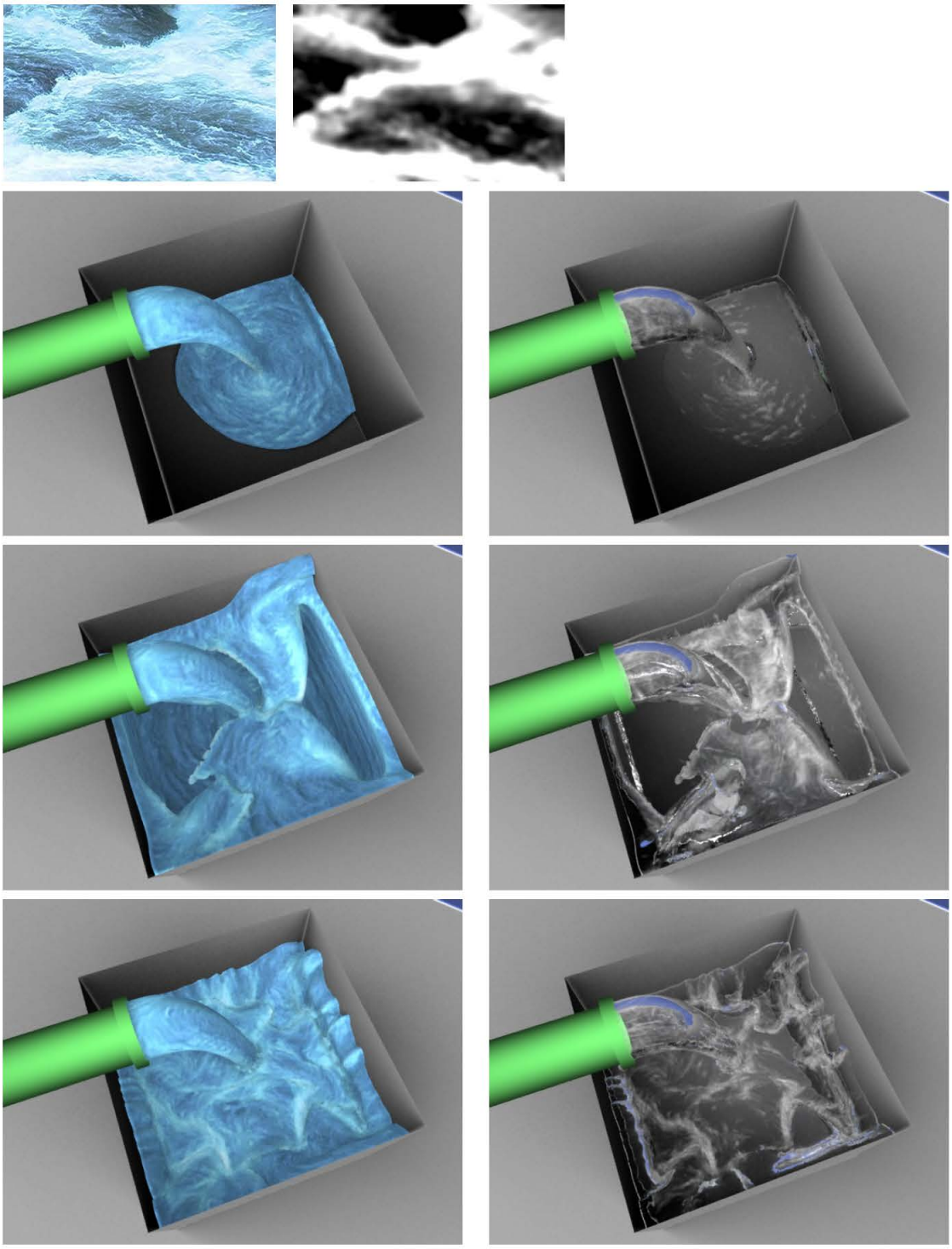
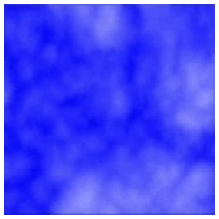


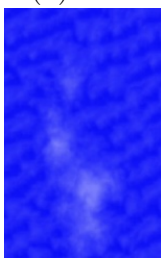
Figure 2.7: Water gushing into a box. The input texture was a video of a turbulent river; one frame is shown at the top. The synthesis output can be used as a diffuse texture (left) or an opacity map (right) for different visual effects.

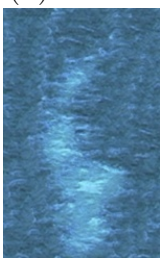

(a) Fire

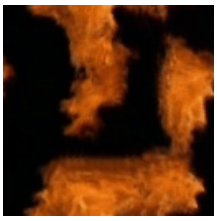

(b) Clouds


(c) River1


(d) River2

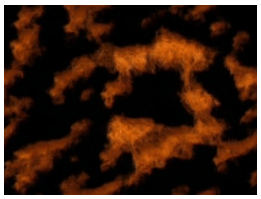

(e) Clouds on Fire

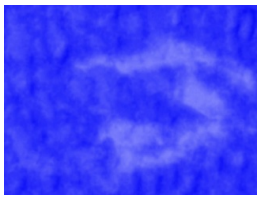

(f) River1 on Fire


(g) Fire on Clouds


(h) River1 on Clouds


(i) River2 on Clouds


(j) Fire on River2


(k) Clouds on River2

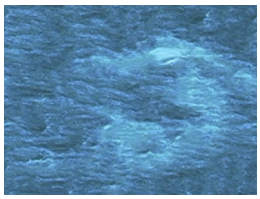

(l) River1 on River2

Figure 2.8: Top row: Frames from the source videos. Middle and lower rows: Results of dynamic texture synthesis to generate new artistic videos using different pairs of input videos. For example, (e) is the result of applying textures from the Clouds video on the features and flow of Fire.

a frame from a $320 \times 320 \times 20$ boiling sequence synthesized using this technique.

2.5.3 DISCUSSION AND LIMITATIONS

The results demonstrate the flexibility of this technique. It can be applied to many different scenarios with only a small amount of user guidance. One limitation of the approach is that if the flow in the video is too fast and complex, then the optical flow computation can be inaccurate. This degrades the quality of the texture synthesis as a lot of the motion is treated as texture evolution and can interfere with the flow in the target scene.

This technique is also computationally intensive, and the unoptimized implementation

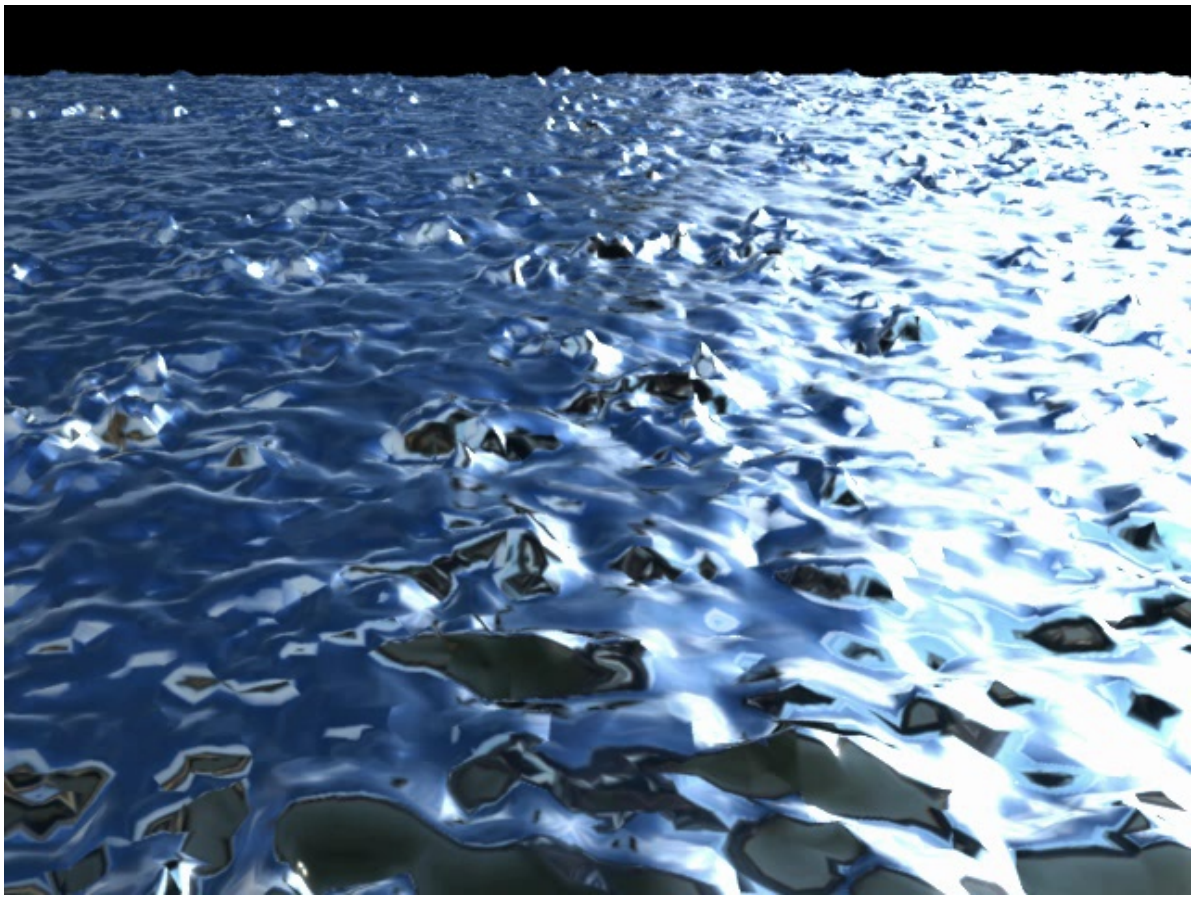


Figure 2.9: Creation of complex liquid surfaces for a boiling water simulation using a texture-based representation. Shown is the surface in one frame from the sequence, obtained as the level set of the synthesized distance field. Input resolution: $64 \times 64 \times 20$. Output resolution: $320 \times 320 \times 20$.

does not incorporate recent work for fast optimization-based texture synthesis, such as that of Han et al. (2006). Depending on the simulation, it typically takes about 200–500 seconds per frame on a 3.4 GHz Pentium 4 processor. Further work could include research on techniques for the acceleration of the algorithm.

CHAPTER 3

TURBULENCE MODELING AND ANIMATION

3.1 INTRODUCTION

Large-scale turbulent fluids such as smoke and water exhibit not just a rich and complex surface appearance, but also visually appealing fine details in their flow. These include swirling vortices, fluctuating eddies, ripples, and other flow patterns that cannot be modeled through textural means. To obtain these fine features in numerical fluid simulation, a mesh with very high spatial resolution is often required for capturing the details, thus the associated computational cost can be prohibitively high. As an alternative, procedural techniques for animating fluids can represent irregular turbulent flow, but require careful animator control and are not always applicable to completely general flow scenarios.

In this chapter, I describe a method for augmenting numerical fluid simulation with a stochastic procedural model for subgrid-scale details to achieve fast visual simulation of turbulent fluids. This turbulence model automatically estimates the intensity of small-scale details induced by the large-scale flow, and introduces swirling eddies and vortices into the fluid using a fast procedural technique inspired by the Kolmogorov law for turbulent flows (Kolmogorov, 1941; Frisch, 1995). The approach presented here has the following characteristics:

- It both compensates for the loss of fine-scale details due to numerical dissipation and introduces additional subgrid-scale flow which cannot be captured on the simulation

grid at all.

- It accounts for the two-way interaction of the procedurally synthesized turbulence with the large-scale simulated flow, enabling the synthesized turbulent flow to affect the dynamics of the numerical simulation for added realism.

- There is only a loose coupling between the fluid solver and the procedural turbulence model, which allows the two to run at different resolutions independently.

- It is general and applicable to a wide spectrum of fluids, including swirling gaseous flows and complex river rapids with free surfaces.

Such an approach can yield very small-scale fluid details from procedural synthesis, while requiring only a low-cost fluid solver on a coarse mesh and achieving an order of magnitude performance gain over high-resolution fluid simulation that produces comparable details. Figure 3.1 shows an animation sequence of roiling smoke around an obstacle synthesized by this method, compared with the result from just the coarse fluid solver. Furthermore, this technique also allows for flexibility in the choice of the numerical fluid solvers. This work has been published as Narain, Sewall, Carlson, and Lin (2008).

3.2 RELATED WORK

The study of turbulence has a long history in the fluid dynamics literature, and remains one of the last open problems in classical mechanics. Seminal work was done in this area by Richardson, Kolmogorov, and Taylor in the early 20th century. The field has seen an enormous body of work since then, but many results remain qualitative, semi-empirical, or restricted to special cases, and a general theory of turbulence remains elusive. Due to the stochastic nature of turbulent fluctuations, a statistical approach has proved useful in this domain (Monin and Yaglom, 1971; McComb, 1990). For nonspecialists, an accessible introduction to turbulence can be found in Davidson (2004).

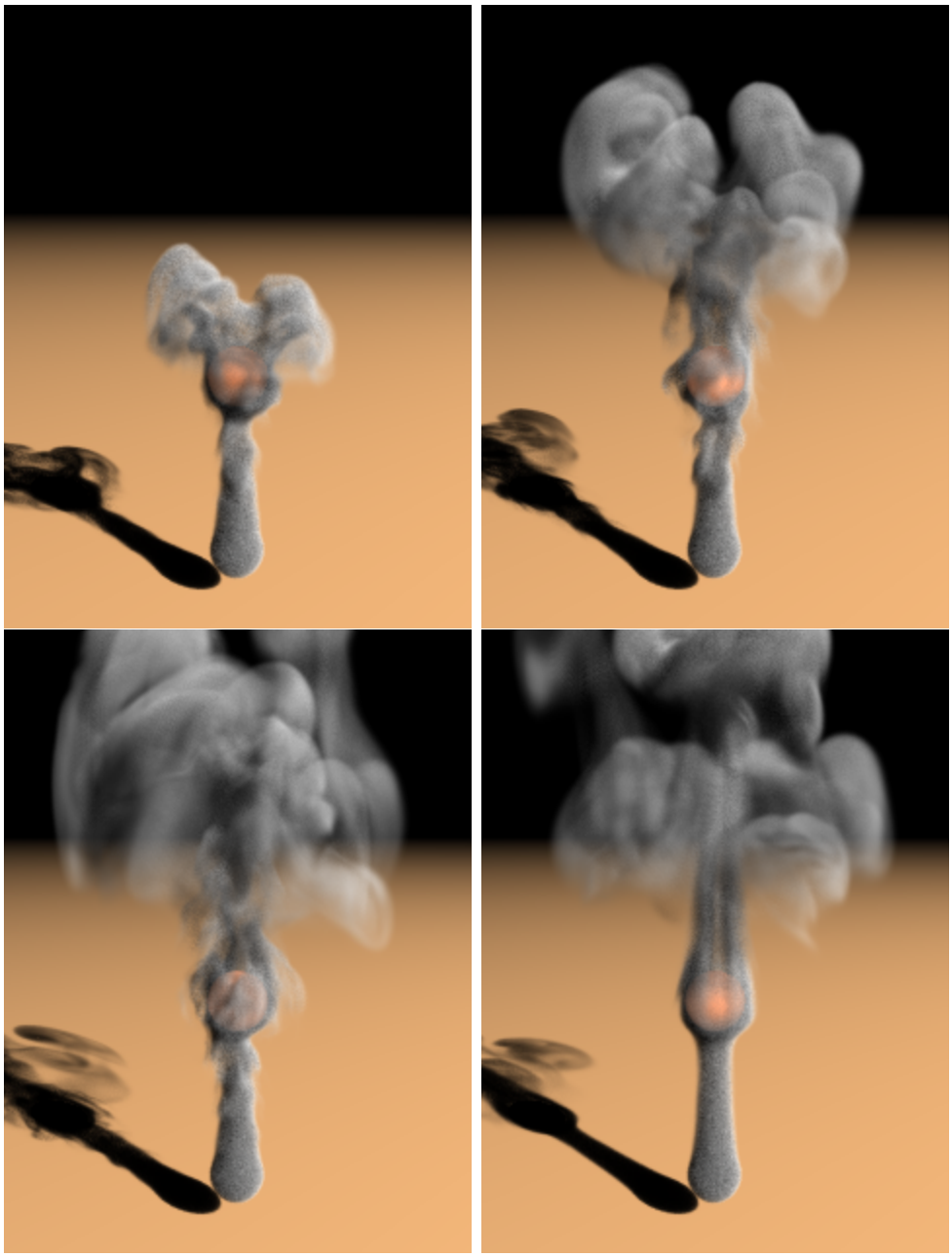


Figure 3.1: Smoke swirls around an obstacle. In reading order: three frames from the result of the combined method, and a corresponding frame from the numerical solver alone.

In computer graphics, physically based animation of fluids such as smoke and water has received considerable attention over the past decade. A popular approach is the use of finite difference methods on Eulerian grids, with semi-Lagrangian methods for advection, as introduced to computer graphics by the pioneering work of Foster and Metaxas (1996); Stam (1999); Foster and Fedkiw (2001). Similar techniques have also been proposed using tetrahedral meshes (Klingner et al., 2006; Chentanez et al., 2007) instead of rectilinear grids. An alternative approach is through Lagrangian methods such as smoothed particle hydrodynamics (Müller et al., 2003; Premoze et al., 2003) which have been used for interactive as well as offline simulation of liquids. Some approaches work with vorticity rather than velocity (Angelidis and Neyret, 2005; Park and Kim, 2005; Elcott et al., 2007). A comprehensive introduction to many of these fluid simulation techniques used in computer graphics can be found in the course notes by Bridson and Müller-Fischer (2007).

One of the prominent difficulties faced in fluid simulation, especially in the popular Eulerian approach, is loss of small eddies and vortices due to numerical diffusion. Consquently, much work has been directed towards avoiding or counteracting this phenomenon to achieve a more detailed, lively appearance in the fluid. Fedkiw et al. (2001) proposed vorticity confinement to amplify small-scale vortices in the fluid. However, if the simulation grid is not fine enough to capture the desired details, vorticity confinement cannot recover them. Selle et al. (2005) discuss this limitation and propose vorticity particles to introduce additional vorticity into the fluid for highly turbulent flows. Their method requires the artist to specify where these particles are injected into the flow, and does not reproduce the decay of turbulence when the forcing is removed; thus, it is best suited for scenarios where a constant source of turbulent vorticity is known *a priori*. Another approach is to use a less dissipative advection scheme (Kim et al., 2007b; Selle et al., 2008; Kim et al., 2008a; Mullen et al., 2009) to reduce the diffusion in the numerical method directly. All of these above methods work directly on the simulation

grid itself, thus none of them can introduce subgrid-scale vortices into the fluid.

Procedural methods for modeling fluid flow are often used by practitioners since they offer cheap evaluation and controllability. Early work synthesized flow fields using a superposition of primitives for laminar flow (Sims, 1990; Wejchert and Haumann, 1991). Turbulent flow was modeled using Fourier synthesis (Shinya and Fournier, 1992; Stam and Fiume, 1993), but this did not allow for spatial modulation of the intensity of turbulence. Recently, curl noise (Kniss and Hart, 2004; Bridson et al., 2007) has been introduced, allowing spatially varying incompressible flow fields that respect rigidly moving boundaries. While curl noise can be used to synthesize a turbulent flow field by adding several octaves of noise, it requires the spatial modulation of turbulence to be specified.

Kim et al. (2008b) used wavelet analysis to determine the characteristics of missing turbulent flow components and synthesize them with band-limited wavelet noise. Another concurrent work by Schechter and Bridson (2008) tracks several bands of turbulent energy using a simple linear model and generates the turbulent velocity using flow noise (Perlin and Neyret, 2001). They also analyzed and corrected the additional vorticity dissipation due to time splitting of the pressure and advection. Their approach is conceptually easy to implement, but requires an animator to seed a distribution of turbulent energy.

3.3 STATISTICAL MODELING OF TURBULENCE

In this section, the mathematical details of modeling fluid flow and turbulence are introduced. The behavior of a viscous, incompressible fluid is described by the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla^2 p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{3.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.2}$$

where $\mathbf{u}$ is the fluid velocity, p is pressure, ν is the kinematic viscosity of the fluid, $\mathbf{f}$ is the external force, and $\nabla = [\partial/\partial x, \partial/\partial y, \partial/\partial z]^T$. The density of the fluid is taken to be

unity.

Fluids obeying the Navier-Stokes equations show smooth laminar flow at low speeds, but as the speed is increased or the viscosity lowered, the flow becomes chaotic and shows irregular fluctuations at large and small scales; this is known as turbulence. Many fluid phenomena of visual interest exhibit a high degree of turbulence, and the range of scales of flow features may span several orders of magnitude. Inevitably, the smallest scales of flow cannot be captured at the resolutions used for practical fluid simulations and are lost due to both insufficient resolution and numerical dissipation.

3.3.1 TURBULENCE AND THE ENERGY CASCADE

Due to the chaotic and stochastic nature of fluid turbulence, a statistical approach is generally adopted both in the theory of turbulence (Monin and Yaglom, 1971; McComb, 1990) and in numerical methods such as Reynolds-averaged Navier-Stokes (RANS) and large eddy simulation (LES). Typically, the fluid flow is separated into a spatially or temporally averaged *mean flow* $\mathbf{U}$ and a fluctuating component $\mathbf{u}'$ where the turbulence resides. While the evolution of $\mathbf{u}'$ itself is extremely chaotic, that of its statistics such as energy is more amenable to modeling.

A popular model for the distribution of energy among different scales is Kolmogorov's famous "five-thirds law" (Kolmogorov, 1941; Frisch, 1995), which states that for homogeneous, stationary turbulence, there is a range of scales in which the distribution of energy $E(k)$ over wavenumber k follows the spectrum,

$$E(k) \propto k^{-5/3} \tag{3.3}$$

This distribution arises due to the *energy cascade* through which turbulent energy is introduced at the *inertial scale* k_0, and cascades to smaller scales until it is dissipated by viscosity. The range of scales over which the Kolmogorov spectrum holds is called the *inertial subrange*, and the characteristics of the turbulent eddies in this range are to a large extent universal and independent of the specific large-scale flow geometry. While the

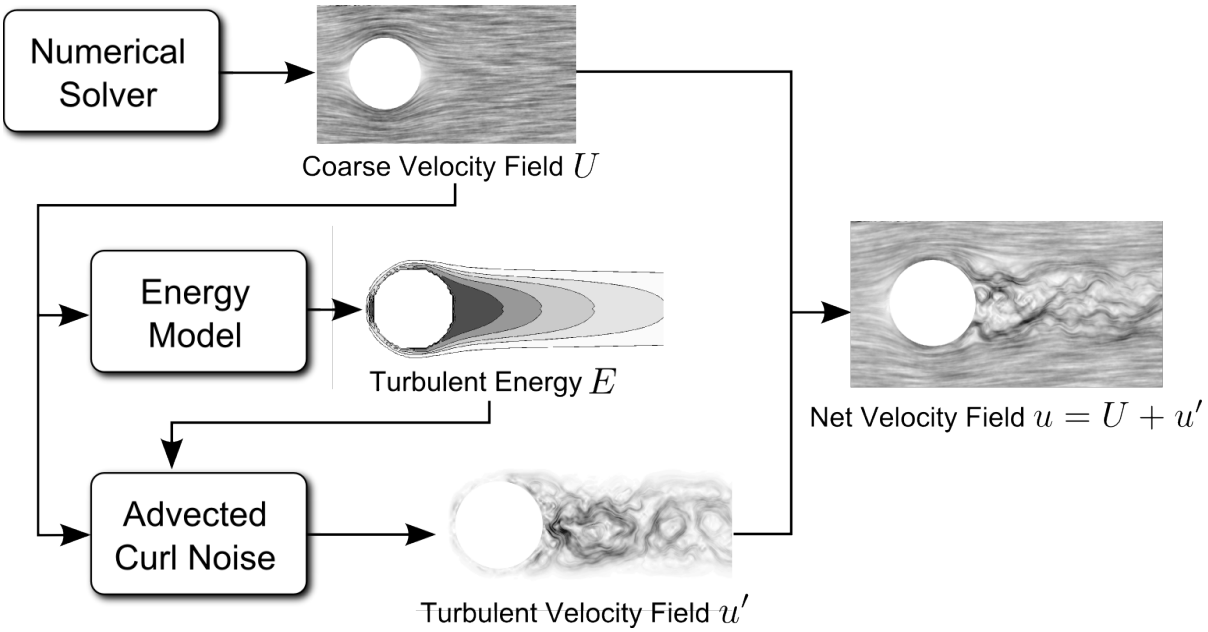


Figure 3.2: The proposed turbulence model applies scale decomposition to combine an existing fluid simulator with an energy model for tracking turbulent energy produced and transported by the mean flow, and an advected curl noise framework for generating the turbulent component of flow.

reverse cascade of energy flow from small to large scales is also possible, it is in practice overwhelmingly dominated by the large-to-small cascade.

3.3.2 OVERVIEW OF THE APPROACH

An effient approach for turbulence modeling can be obtained through the similar idea of scale decomposition, describing the velocity field $\mathbf{u}$ of the fluid as the sum of a large-scale flow $\mathbf{U}$ and a small-scale turbulent component $\mathbf{u}'$. In the method I describe, $\mathbf{U}$ is taken to be the velocity computed by an existing numerical simulator, which sufficiently resolves the important large-scale dynamics of the flow. An explicit model for small-scale fluctuations $\mathbf{u}'$ is introduced. By construction, the sub-grid kinetic energy associated with $\mathbf{u}'$ falls off with decreasing scale in a way similar to what is observed in the inertial subrange of isotropic turbulence.

The small-scale turbulent flow can be characterized in terms of the distribution of its

kinetic energy over space and over different scales. The two main parts of this technique, shown in figure 3.2, are (1) tracking the production, transport and dissipation of this energy over time due to the large-scale flow $\mathbf{U}$, and (2) synthesizing a small-scale turbulent flow field $\mathbf{u}'$ which obeys this energy distribution. The fine-scale flow generated is used to produce highly detailed output data such as billowing smoke or fine ripples on the free surface of a liquid, and is also loosely coupled to the large-scale flow $\mathbf{U}$, introducing vortices and other flow detail at the small grid scales resolved by the simulation.

3.4 DYNAMICS OF TURBULENT ENERGY

To make the problem approachable, let us start with the simplifying assumptions that the turbulence is isotropic, follows the Kolmogorov spectrum at each point, and has local eddies which are uncorrelated with the large-scale flow. Dividing the spectrum into octaves of spatial frequency in order of decreasing length scale $l_i = l_{i-1}/2$ and increasing wavenumber $k_i = l_i^{-1}$, the quantity to be determined is the kinetic energy density E_i. By the Kolmogorov spectrum, the kinetic energy E_0 in the lowest octave of turbulent eddies fully defines the energy in the ith octave simply as

$$E_i = E_0(k_i/k_0)^{-2/3} \tag{3.4}$$

The change in exponent from $-5/3$ to $-2/3$ from (3.3) is because here E_i describes the energy in one *octave*, which is obtained by integrating the energy per unit *wavenumber* over an octave range. The quantities E_i are allowed to be spatially varying scalars in order to represent the distribution of turbulence in complex inhomogeneous flow. The magnitude of the eddy velocities at any scale are then on the order of $\bar{u}_i = \sqrt{E_i}$, which will be directly used to generate a spatially modulated velocity field.

There is no closed solution for describing the behavior of the turbulent kinetic energy. Nevertheless, on physical grounds, the important physical phenomena involved in the energy dynamics of turbulence are as follows: (1) production of turbulence via forcing

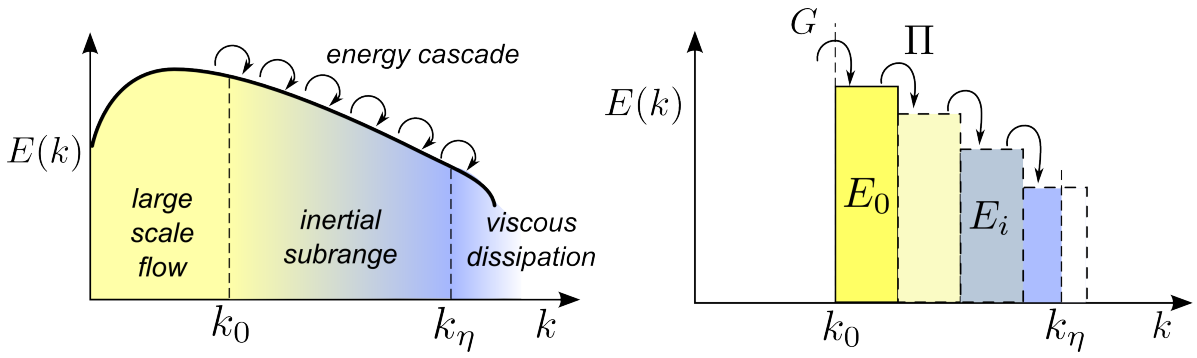


Figure 3.3: The turbulent energy cascade (left) transports energy from the mean flow through the large scales of turbulence to the smallest ones. In the discretized model (right), only the energy E_0 in the largest turbulent band is tracked, and the rest is modeled using the Kolmogorov spectrum.

from the mean flow, (2) advection by the mean flow, (3) diffusion of kinetic energy due to turbulent mixing, (4) cascade of energy from larger to smaller scales due to non-linear self-advection, and (5) viscous dissipation, which removes energy at the smallest scale of eddies. Thus, in abstract terms, the dynamics of energy at the inertial scale can be expressed as

$$\frac{\partial E_0}{\partial t} + \mathbf{U} \cdot \nabla E_0 = G(E_0, \mathbf{U}) - \Pi(E_0) + D(E_0) \tag{3.5}$$

where G is the production term through which turbulent energy is generated, Π is the rate of energy flow through the cascade, and D describes the spatial diffusion of energy due to turbulent mixing. Dissipation is ignored since viscosity is only significant at the smallest scales. Figure 3.3 shows how this approach relates to the theoretical energy cascade.

If these terms are estimated well, the behavior of the turbulent eddies will behave in a physically plausible way. I now describe approximations for the magnitudes of these quantities to achieve this.

3.4.1 ENERGY TRANSFER

First, the inertial scale k_0 must be chosen to define the size of the largest turbulent eddies modeled. In practice, one can choose the eddy size $l_0 = k_0^{-1}$ to be a small multiple of the

grid spacing of the numerical simulator. Overlapping the scales modeled numerically and procedurally in this way has two advantages: it allows the turbulent flow to visually "fill in" flow details lost to numerical diffusion, and it makes it possible for the turbulence to affect the numerically simulated flow at the overlapping scales.

The production term Through G, the mean flow acts as a driving force which creates and intensifies turbulent energy. The simplest approximation that can be made here is that turbulence acts as a viscous drag on the mean flow, characterized by an *eddy viscosity* ν_t and taking energy at the rate $\nu_t S(\mathbf{U})^2$, where $S(\mathbf{U})^2$ is the squared norm of the strain rate, defined below. The eddy viscosity can further be approximated through mixing-length theory as $\nu_t = l_0 \bar{u}_0 = k_0^{-1}\sqrt{E_0}$ (Davidson, 2004). This gives the net rate of energy production as

$$G(E_0, \mathbf{U}) = k_0^{-1}\sqrt{E_0} S(\mathbf{U})^2 \tag{3.6}$$

with $S(\mathbf{U})^2 = \sum_{i,j} \left((\partial U_i/\partial x_j + \partial U_j/\partial x_i)/2\right)^2$.

The cascade term A simple estimate can be obtained using the conjecture of Obukhov (via Davidson (2004)) that turbulent eddies at the inertial scale pass on energy into smaller ones on the timescale $\tau_0 = l_0/\bar{u}_0$. Since the energy carried by these eddies is E_0, the rate of energy transfer becomes

$$\Pi(E_0) = E_0/\tau_0 = k_0 E_0^{3/2} \tag{3.7}$$

The diffusion term Kinetic energy at a single octave diffuses over space as turbulent eddies transport fluid, causing momentum and energy transfer. By analogy with the turbulent mixing of a passive scalar being advected by the turbulent flow, whose diffusion rate is roughly $\alpha \propto l_0 \bar{u}_0$ (Davidson, 2004), a natural approximation for the turbulent diffusion of energy as

$$D(E_0) = \nabla \cdot (\alpha \nabla E_0) = \nabla \cdot (l_0 \sqrt{E_0} \nabla E_0) \tag{3.8}$$

Viscosity The role of viscosity in turbulent flow is only significant at the smallest eddy scales, and is limited to dissipating the cascading energy at the highest wavenumbers. To reproduce this, the energy spectrum is simply cut off at the scale where the rate of viscous dissipation begins to dominate the cascade rate Π. This *Kolmogorov microscale* k_ν is given by (McComb, 1990)

$$k_\nu = \left(\frac{\Pi(E_0)}{\nu^3}\right)^{1/4} \tag{3.9}$$

After the energy E_0 at the largest octave is known, the energy E_i at any other octave i with reciprocal length scale k_i is determined using a Kolmogorov spectrum cut off at k_ν, as figure 3.3 illustrates. If the cutoff falls within an octave, that octave's energy is scaled by how much of it lies before the cutoff. This approach allows the model to "turn off" turbulence appropriately when the flow is calm, as k_ν will become lower than the first octave k_0 itself, and no energy will be present in the subgrid scales.

$$E_i = \begin{cases} E_0 \left(\frac{k_i}{k_0}\right)^{-2/3} & \text{if } 2k_i < k_\nu \\ E_0 \left(\frac{k_i}{k_0}\right)^{-2/3} \frac{2k_i - k_\nu}{2k_i - k_i} & \text{if } k_i \leq k_\nu \leq 2k_i \\ 0 & \text{if } k_\nu < k_i \end{cases} \tag{3.10}$$

3.4.2 NUMERICAL ISSUES

The dynamics of turbulent kinetic energy as given in (3.5) essentially form an advection-reaction-diffusion PDE. Following Kim and Lin (2007), the integration of this PDE is split into diffusion, advection and reaction stages, allowing stability to be maintained at each step.

Since the turbulent energy E_0 is governed by the large-scale flow $\mathbf{U}$ from the numerical method, it is represented on the same coarse grid. This allows the same large timesteps to be taken as the numerical simulation. The advection step can be performed using the same scheme as used in the fluid solver.

For the diffusion step, the nonlinearity in the diffusion rate presents numerical difficulties at regions where E_0 has large changes in magnitude. In the interest of stability and

efficiency, it can be replaced with a constant spatially varying term which is filtered to avoid the degeneracy, as follows:

$$\tilde{D}(E_0) = k_0^{-1} \nabla \cdot \left(\sqrt{\tilde{E}_0^t} \nabla E_0^{t+1} \right) \tag{3.11}$$

where $\tilde{E}_0^t$ is a filtered version of E_0^t obtained by Gaussian filtering with a radius l_0. This spatially varying but linear diffusion equation can then be integrated efficiently by an alternate dimensions implicit (ADI) method (Kass et al., 2006). This provides a fast approximate solution which is unconditionally stable, and acceptably accurate as long as $\overline{u_0} < l_0/\Delta t$.

The turbulence production and cascade terms require no neighbor information, and so can be solved as a simple pointwise ODE, with $E_0' = G(E_0, \mathbf{U}) - \Pi(E_0) = f(E_0)$. This ODE has a unique stable equilibrium for positive E_0. Large timesteps can be taken with stability by choosing between forward Euler or backward Euler timestepping based on the sign of df/dE_0.

The initial value of E_0 should not be precisely zero since the production term G contains a factor with E_0 itself. However, in practice, a very small value such as 10^{-4} can be chosen; this does not affect the characteristics of the results, which are dominated by the interaction of the production and cascade terms.

A note on boundary layers Physically, a solid boundary immersed in fluid imposes a no-slip condition on the flow, producing a thin boundary layer over which the relative velocity of the fluid decreases to zero. It is here that turbulence due to obstacles is generated. Explicitly accounting for such boundary layers would add significantly to the implementation complexity of the model, and the extremely high strain rate and extremely small thickness of the layer could lead to numerical difficulties. A simpler alternative is to account for the no-slip condition by using a ghost velocity past the boundary equal that of the obstacle when computing $\mathbf{S}(\mathbf{U})$. Subsequent work by Pfaff

et al. (2009) has presented an efficient approach to approximating the boundary layers using data gathered from high-resolution precomputations.

3.5 PROCEDURAL SYNTHESIS OF TURBULENT FLOW

Having defined the distribution of turbulent energy over space and on different scales, it remains to actually generate an incompressible, turbulent velocity field $\mathbf{u}'$ which has the prescribed energy distribution. The eddies and vortices in this velocity field must also move with the large-scale flow, and fluctuate over time for a realistic appearance of turbulence. In this section, I describe a method to generate such a velocity field which has these properties.

Curl noise (Kniss and Hart, 2004; Bridson et al., 2007) uses the vector calculus identity $\nabla \cdot (\nabla \times \psi) = 0$ to provide an efficient means of generating incompressible flow fields supporting spatial modulation and rigid boundaries. Several octaves of such curl noise can be added, each modulated by the desired velocity magnitude $\overline{u_i}$, to generate an instantaneous velocity field with the desired energy spectrum. However, the temporal behavior of the velocity field over time is also important. First, the turbulent velocity field should be advected by the mean flow $\mathbf{U}$, in order for the turbulent eddies to be seen as being carried along by the fluid. Second, the turbulent velocities should also fluctuate over time at a rate based on the eddies' characteristic timescale, given by τ_i which varies over space.

Neyret (2003) has presented a method for advecting and regenerating textures to avoid distortion; this has been used in concurrent work by Kim et al. (2008b) for advecting wavelet turbulence. In experiments, I found that textures had to be regenerated too often in regions of high strain rate, and the use of texture coordinates made looking up τ_i difficult. I propose a simpler Lagrangian approach, representing the potential function using basis functions centered on particles which are advected with the flow. Such noise particles had been used previously by Stam and Fiume (1993) for advecting smoke, and

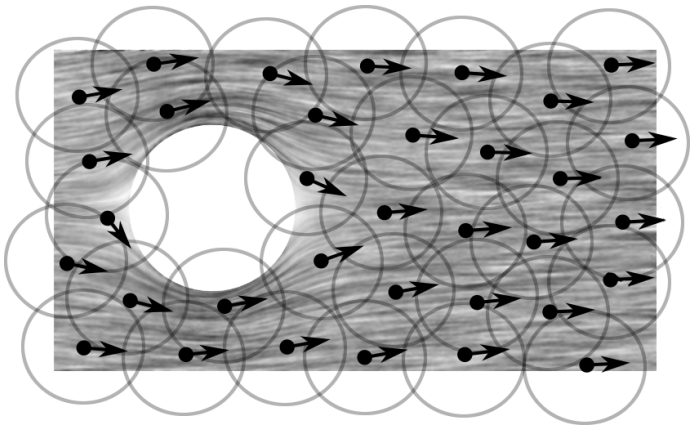


Figure 3.4: Noise particles are distributed uniformly in the fluid volume and advected by the large-scale flow.

Zhu and Bridson (2005) for gritty sand rendering. In combination with curl noise, such an approach allows for efficient, distortion-free transport of incompressible vector noise.

3.5.1 NOISE PARTICLES

In this approach, a vector potential field ψ is represented as the sum of volumetric noise textures centered at a set of particles $\mathbf{p}_j$ distributed uniformly within the fluid, as illustrated in figure 3.4. A smooth radial falloff $\rho(r)$ is used for blending between particles. For noise at a particular scale i, each particle carries an animated vector noise function $\mathbf{N}_j^i(\mathbf{r}, t_i)$ of the appropriate scale. The net potential for one octave, ψ_i, at a point $\mathbf{x}$, modulated by a spatially varying amplitude A_i, is then given by the weighted average of scaled noise functions,

$$\psi_i(\mathbf{x}) = \frac{\sum_j \rho(\|\mathbf{x} - \mathbf{p}_j\|) A_i(\mathbf{p}_j) \mathbf{N}_j^i(\mathbf{x} - \mathbf{p}_j, t_i)}{\sum_j \rho(\|\mathbf{x} - \mathbf{p}_j\|)} \tag{3.12}$$

Moving boundaries are accounted for by further modulating ψ_i using the approach of Bridson et al. (2007). The derived velocity field is simply $\mathbf{u} = \nabla \times \psi = \nabla \times \sum_i \psi_i(\mathbf{x})$.

For the radial falloff $\rho(r)$, one requires a kernel which goes smoothly to zero at $r = l_0$ and has continuous derivatives; the cubic spline $1 - 3(r/l_0)^2 + 2(r/l_0)^3$ suffices. For different octaves, the amplitude is chosen as $A_i = \overline{u_i} l_i$ so that the derived velocity field

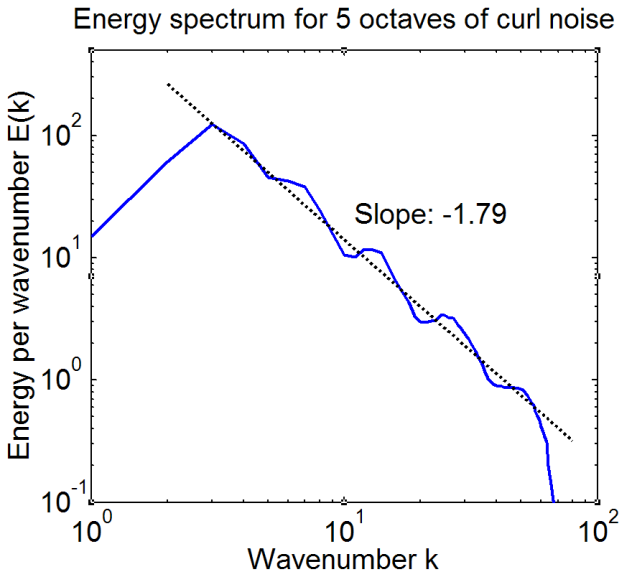


Figure 3.5: The energy spectrum of turbulent flow generated by 5 octaves of curl noise shows a power law with exponent close to the expected value of -1.66.

has magnitude $\overline{u_i}$. The three components of the vector noise functions $\mathbf{N}^i$ are taken as different offsets of animated 3D Perlin noise, which is precomputed. Wavelet noise (Cook and DeRose, 2005) could be used for better band-limiting and isotropy characteristics, as suggested by Kim et al. (2008b). It was found that the use of Perlin noise nevertheless has reasonable spectral behavior as shown in figure 3.5, and does not harm the visual plausibility of the result.

Initially, the particles are distributed over the volume of the fluid in a Poisson-disk distribution, with a distance of l_0 to allow the RBFs ρ to overlap. Advection from the mean flow $\mathbf{U}$ is taken into account by simply transporting the particles forward with the flow. This has the effect of transporting the derived velocity field $\mathbf{u}$ along the flow as well, while also maintaining incompressibility and avoiding anisotropic distortion which would not have been the case had $\mathbf{u}$ been advected directly. A dynamic sampling scheme could also be used to delete and insert particles to maintain an approximate Poisson-disk distribution in 3D (Bridson, 2007), but this was found not to be necessary in practice.

It remains to make the generated velocity field exhibit temporal fluctuations on the

timescale of the eddy turnover time τ_i. Recall that each particle carries an index into an animated noise function, whose timescale can be chosen such that it fluctuates, say, once every unit t. Then the time coordinate t_i of the noise function can be advanced by τ_i^{-1} per second, which has the desired effect.

3.5.2 COUPLING TO THE LARGE-SCALE FLOW

Finally, I describe the method by which the generated turbulent flow affects the large-scale fluid flow handled by the fluid simulator. Unlike Selle et al. (2005), since here the turbulent flow represented by particles and the large-scale simulated flow are kept separate, the vorticity confinement force is not used to drive the simulated fluid. Instead, the turbulent velocity field $\mathbf{u}'$ simply acts as an additional velocity when performing the self-advection step in the simulation. That is, the simulation velocity $\mathbf{U}$ is advected not just by itself but by the sum $\mathbf{U} + \mathbf{u}'$, appropriately resampled to the simulation grid. This simple coupling models the visually important effects of turbulent advection on the coarse grid.

Similarly, scalar quantities such as density for smoke animations are also advected by the net velocity field $\mathbf{U} + \mathbf{u}'$. These scalar quantities and the final turbulent velocity are represented on finer grids than the simulation, allowing high-resolution detail to be obtained without increasing the computational cost of the numerical simulation. The density has to be downsampled at each step to compute the forcing effects such as buoyancy forces on the simulation, but this is a cheap operation. Since it is these scalar quantities that are directly visualized and not the fluid velocity field itself, this approach provides a direct visual benefit.

3.5.3 FREE SURFACES

A similar approach can be applied to the animation of liquids, where by representing the free surface at high resolution, fine surface details such as small ripples can be obtained. However, a fundamental issue is that in the presence of a free surface, the isotropy and

homogeneity assumptions of the Kolmogorov model break down, and the turbulent surface features show a complex dynamical behavior for which "the present state of the art is quite controversial" (Magnaudet, 2003). In the absence of a single well-accepted theory of free-surface turbulence, a reasonable compromise for visual applications is to extend the Kolmogorov-based model that maintains the coarse simulation grid and synthesizes the fine-scale behavior procedurally, even though this falls well outside the model assumptions. This is presented as initial work on the topic of free-surface turbulence for computer animation, in the hope that it will encourage further exploration.

First of all, since the free surface also defines the simulation domain for the numerical method, a conservative downsampling is necessary to ensure known values of fluid velocity throughout the fluid. A simulation cell is marked as fluid if any fluid is present in it in the high resolution surface. To take into account the free surface fluctuations, it can be modeled as a variable density flow, with the density equated to the filled volume fraction. For this, the pressure projection equation is modified to be $\mathbf{U} = \mathbf{U}^* - (\nabla p)/\rho$, and therefore $\nabla \cdot \mathbf{U}^* = \nabla \cdot (\nabla p/\rho)$ as described in Kim et al. (2007b), where $\mathbf{U}^*$ is the pre-projection velocity field.

Another problem is that the simulation is unaffected by the high-resolution ripples introduced on the fluid surface. This means that subgrid-scale surface ripples have no dynamical effect and do not die out over time, making the fluid appear to be viscoelastic. This can be counteracted by introducing two additional effects to make the high-resolution ripples, if present, continue to move and fluctuate, and gradually die out over time. These effects do not exactly correspond to the equations of wave dynamics, however. Attempting to simulate the wave nature of subgrid-scale ripples would tie the timestep to the very small grid size of the high-resolution surface, which is something to be avoided for fast simulation.

First, if the surface is represented as the level set of a function ϕ, one can obtain a smoothed version $\tilde{\phi}$ in an efficient manner by filtering and downsampling it to the coarse

simulation grid, and reinitializing it by a standard fast marching method, such as that described by Losasso et al. (2005). The smoothed level set $\tilde{\phi}$ represents a surface with subgrid-scale features removed. The decay of the small-scale features can be modeled by timestepping ϕ to approach $\tilde{\phi}$ over a timescale τ'.

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{\tau'}(\tilde{\phi}^n - \phi^n) \tag{3.13}$$

The characteristic timescale τ' of subgrid features is defined in terms of the simulation grid resolution Δx as $\tau' = \Delta x/c$, where $c = \sqrt{g\Delta x/(2\pi)}$ is the speed of deep water gravity waves of wavelength Δx. Surface tension only becomes significant for waves of wavelength smaller than 1.7 cm (Lamb, 1993); this is neglected in the current work.

Secondly, the characteristic "rippling" motion can be obtained by ensuring that there is always some turbulent energy associated with the high-resolution features which will cause them to move and fluctuate. The surface turbulent energy associated with the ripples is defined as

$$E' = \overline{u'}^2 = (\delta/\tau')^2 \tag{3.14}$$

where δ is the average height of the ripples. δ can be estimated as the average magnitude of $\phi - \tilde{\phi}$ in the grid cell. Using $\max(E_0, E')$ instead of E_0 as the turbulent energy distribution at each point, plausible motion of ripples is achieved in a simple and efficient manner.

3.6 RESULTS AND DISCUSSION

The technique described in this chapter was implemented using a particle level set solver (Bridson and Müller-Fischer, 2007) with BFECC (Kim et al., 2007b) for the numerical simulation of the fluid. In all scenarios, a $4 \times 4 \times 4$ times finer resolution was used for the turbulence as compared to the simulation grid resolution for the fluid solver. This 4x refinement was found to be adequate to resolve visually relevant details in most scenarios; higher values would produce finer detail (as long as the intensity of turbulence is high enough) but would not otherwise change the characteristics of the result.

The primary control parameter which defines the behavior of the fluid is its viscosity ν. This controls the viscous cutoff k_ν in equation (3.10). Higher values would delay the onset of turbulence and prevent very fine eddies from forming, as appropriate for a more viscous fluid. The other parameter that can be varied is the size of the largest eddies, l_0. This parameter controls the scale of the injected turbulent flow. It is typically be set at 4 times the size of the simulation grid cells, but can be changed for more artistic control. For example, the rocket launch example reduced it by half to obtain relatively smaller eddies.

3.6.1 BENCHMARKS

I present several examples of results from the presented technique. Figure 3.6 shows smoke rising from a spherical density source due to buoyancy forces. The smoke column is only a few grid cells across, and BFECC is not able to recover any detailed vorticity. The proposed turbulence model produces visually plausible results and reproduces the transition from laminar to turbulent flow as flow velocity increases, thanks to the viscous cutoff at k_η. This would be difficult to achieve with a vortex particle method. A comparison is also shown with the results of vorticity confinement (Fedkiw et al., 2001). On the original low-resolution grid, it had little effect; a simulation on a 2x refined grid is shown, which exhibits turbulent behavior but is biased towards only the finest-scale eddies.

Free surface handling is demonstrated in figure 3.7. The presented method captures very high-resolution features in the wake of the moving ball, which appropriately die down over time as the driving mean flow is removed. More free-surface flows are shown in the water pouring into box (figure 3.8) and river rapids flowing over the bed rocks and tree logs (figure 3.9). In all cases, the results are rendered with the high-resolution surface while the base simulation has only the coarse level set.

In figure 3.10, the turbulent exhaust jet during a rocket launch is simulated. The

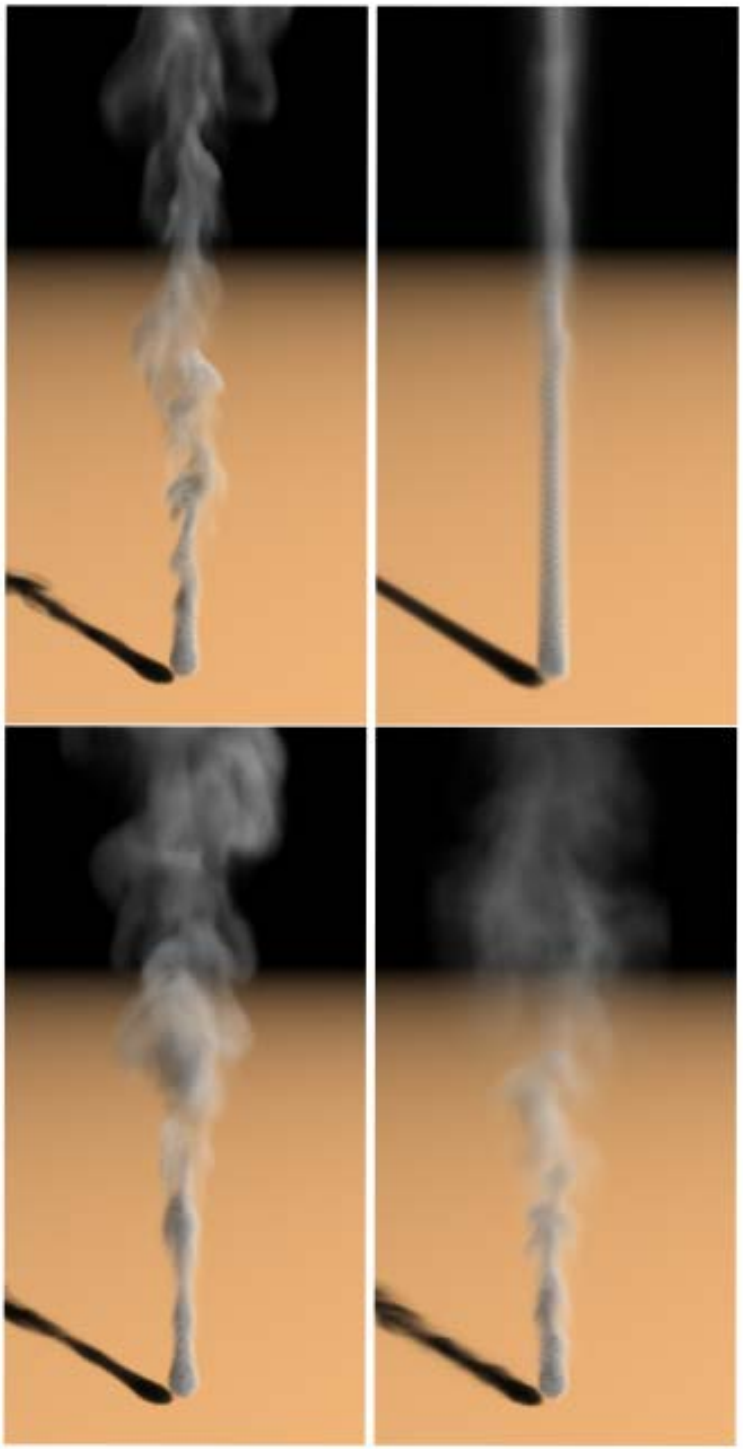


Figure 3.6: Smoke rising from a small density source, simulated with (a) the turbulence model, (b) low-resolution simulation only, (c) reference simulation on a 4× refined grid, and (d) vorticity confinement on 2× refinement.

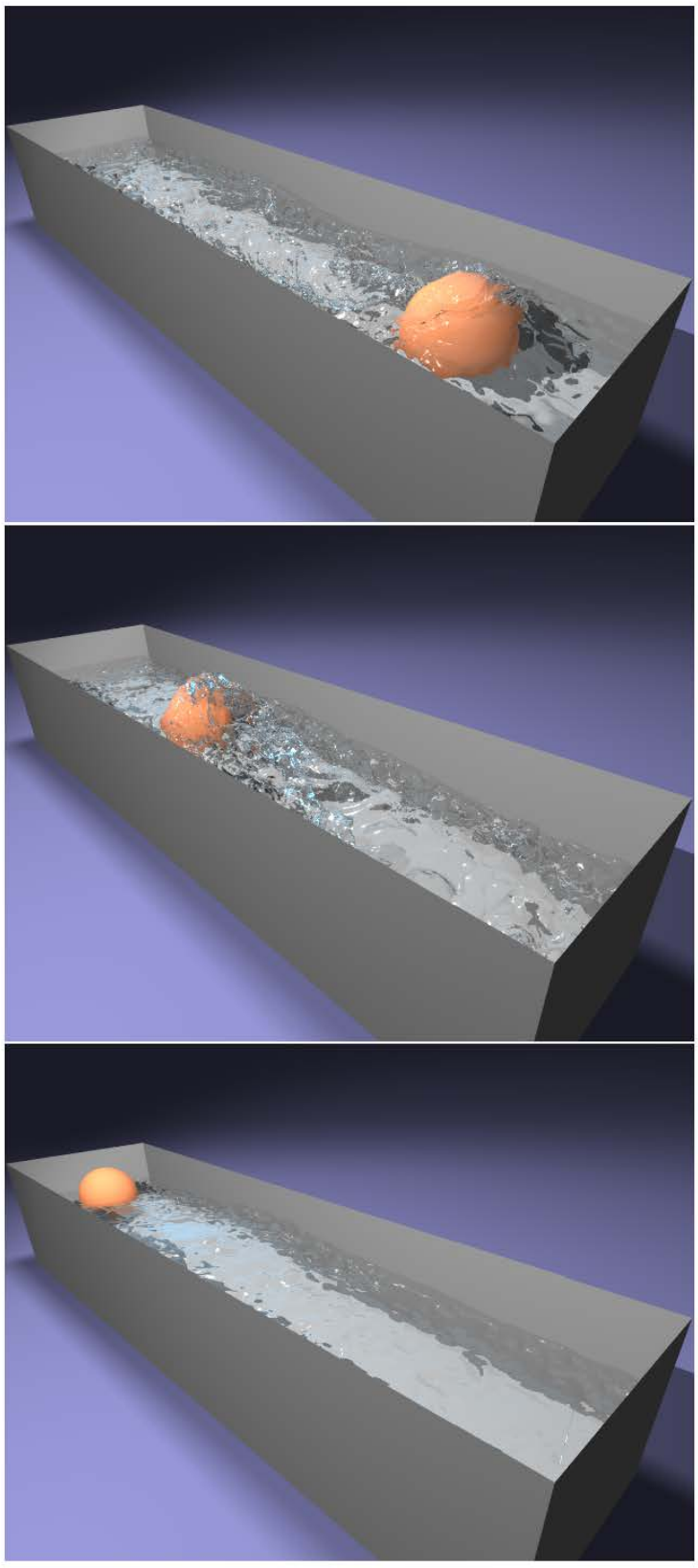


Figure 3.7: A moving ball creates a turbulent wake behind it.

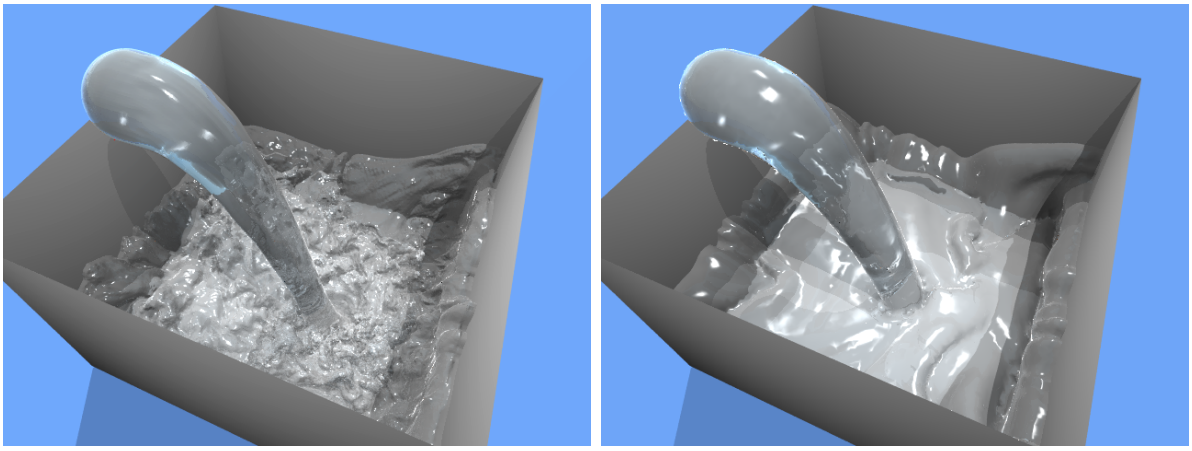


Figure 3.8: Water pouring into a tank creates a chaotic, turbulent surface. In this and later figures, the results using scale decomposition on the left show fine surface details, while the results of fluid simulation without added turbulence, on the right, can only generate a very smooth fluid surface.

Scene	Grid size	Ours	High-res	Gain
Smoke (Fig. 3.6)	$256 \times 32 \times 32$	56 sec	1192 sec	21x
Wake (Fig. 3.7)	$128 \times 32 \times 32$	45 sec	708 sec	16x
Box (Fig. 3.8)	$64 \times 64 \times 64$	70 sec	360 sec	5x
Rapids (Fig. 3.9)	$150 \times 30 \times 50$	105 sec	776 sec	7x
Launch (Fig. 3.10)	$64 \times 64 \times 64$	142 sec	3186 sec	24x

Table 3.1: Performance comparison of the proposed method with a full high-resolution simulation needed to generate the same visual detail without added turbulence. All timings are per-frame and include the time taken by the fluid solver, the turbulence model, and scalar advection.

turbulent eddies show plausible motion in both the jet itself and the billowing smoke on the ground. The rocket launch simulation used 98,000 noise particles due to the choice of relatively smaller l_0. All other simulations used between 6,1000 and 12,300 noise particles.

3.6.2 PERFORMANCE AND COMPARISON

The proposed turbulence model is an order of magnitude faster than a full numerical simulation at 4x higher resolution. The time for tracking the energy distribution was negligible. The bottleneck in the approach turned out to be the evaluation of the potential in advected curl noise, because this step involves irregular memory accesses on a very

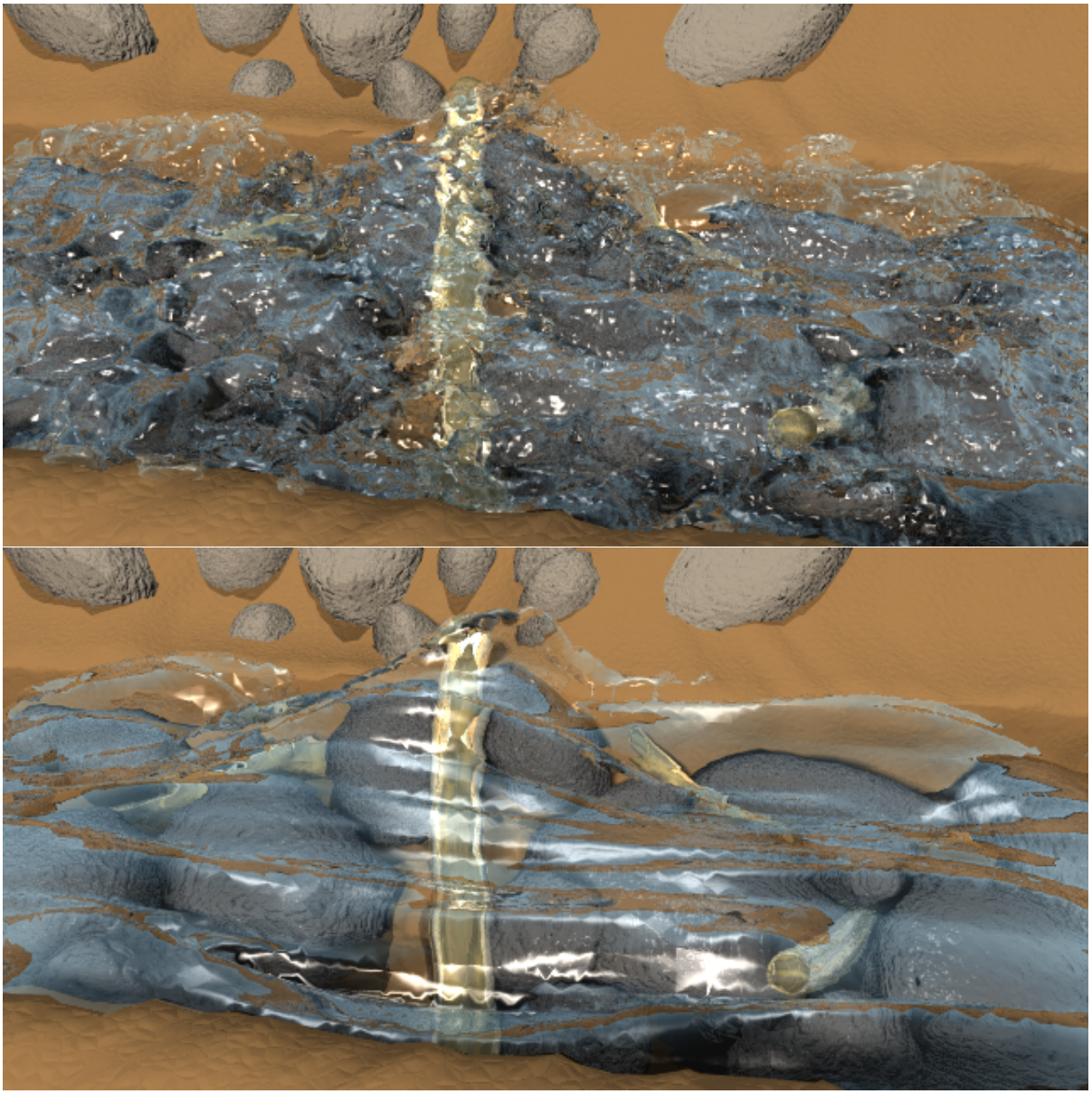


Figure 3.9: River rapids show turbulence due to fast-moving flow in a complex channel with many obstacles.

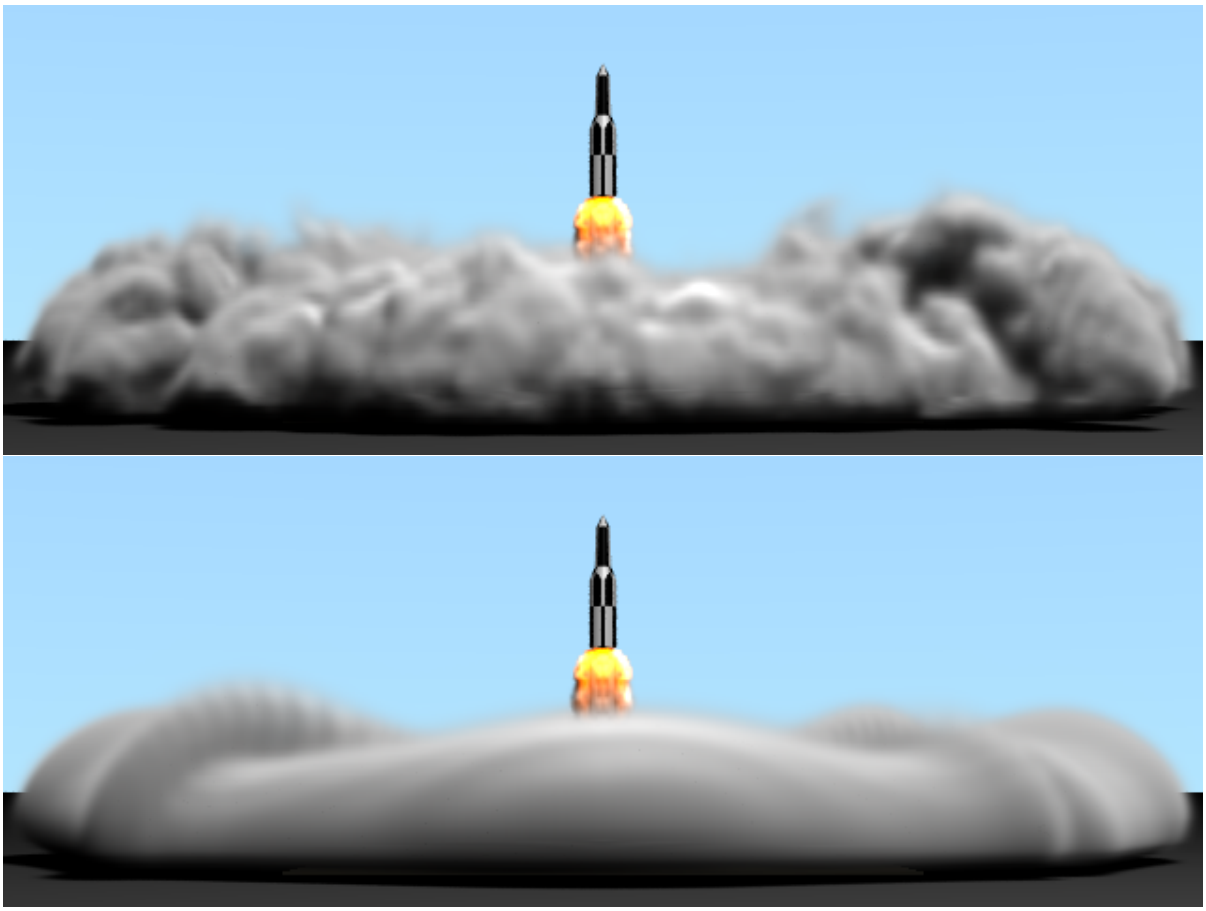


Figure 3.10: Simulation of turbulent exhaust jet during a rocket launch.

high-resolution grid. Timings were taken on an Intel Xeon 2.8 GHz processor with 8 GB of RAM. Table 3.1 summarizes the performance of the method in comparison with basic low- and high-resolution simulations. Liquid simulations with high-resolution advection alone were not performed because of the free-surface issues discussed in section 3.5.3.

3.6.3 ANALYSIS AND DISCUSSION

The results demonstrate the flexibility and generality of this method. It replaces the computational complexity of high-resolution fluid simulation with a simple turbulence model, giving a major performance benefit since fluid simulation on a coarser grid can take much larger timesteps than possible at the full fine-scale resolution. The turbulence model itself is stable and can take equally large timesteps as the coarse fluid simulation

allows. There is only one global solve, which appears in the diffusion step and is performed efficiently through dimensional splitting.

The main bottlenecks in speed were the evaluation of the noise function, and the maintenance of the free surface. A comparison of the proposed advected noise algorithm with that of Neyret (2003) in terms of quality and performance is left for future work. For the free surface, the level set needs to be reinitialized at each step, which causes a performance hit.

Interesting questions remain about how much the scales of procedural turbulence and the simulated flow should overlap. Less dissipative numerical simulations as proposed in recent work would likely not need as much fluctuations to be injected at the grid level, so l_0 might be chosen to be as small as the grid size. Further, whether the combined spectrum of the large-scale and turbulent flow has an appropriate shape may be worth investigating. These experiments are left for further investigation.

Finally, the applicability of the Kolmogorov model depends on the assumptions of isotropy and homogeneity, and as such, the model is being extended beyond its strict bounds to model the complex, directed flows seen in animation. The surface handling approach is relatively simplistic and cannot guarantee volume conservation. However, the complexity of the phenomenon of turbulence is such that physical accuracy is unachievable without great computational expense. The approach presented here was deliberately chosen to approximate the behavior of turbulent fluids, so that a simple and efficient technique is possible for generating plausible visual effects. Much research is still needed in modeling anisotropy, higher-order statistics, detailed free-surface phenomena, and other properties of true physical turbulence without an exorbitant computational cost.

CHAPTER 4

SIMULATION OF DENSE CROWDS

4.1 INTRODUCTION

In this and the following chapter, the focus turns from fluids to the flow of large aggregates, such as crowds of pedestrians, and piles of sand grains. Unlike fluids, which are to all practical purposes *continuous* media, these aggregate systems are composed of numerous *discrete* entities interacting with each other. Nevertheless, large systems of this kind often exhibit a smooth, coherent collective flow on a coarse level which appears continuous in nature, in contrast to the irregular and unpredictable motion of individual objects at the fine scale. These different characteristics of behavior call for a scale decomposition approach to the problem, combining a continuous representation of a macroscopic flow with a discrete model of individual objects at the microscale. In this chapter, such an approach is presented for efficient crowd simulation. The model for granular materials described in the next chapter builds upon much of the principles developed here.

Crowds are ubiquitous in the real world, and simulating their motion in virtual environments is an important topic in computer graphics. Crowd simulation techniques find application in diverse fields such as training, entertainment, education, architecture, and urban planning. Human crowds exhibit highly complex behavior driven by individual decisions of agents with respect to their goals, environmental obstacles, and other nearby agents. Existing approaches often decompose this problem into global planning from

local collision avoidance. The local collision avoidance module requires each agent to take into account the motion of its nearby neighbors; this step quickly turns into a major computational bottleneck for very dense crowds.

Here, I focus specifically on the problem of simulating the inter-agent dynamics of large, dense crowds in real time. Such crowds exhibit a low interpersonal distance and a corresponding loss of individual freedom of motion. This observation suggests that the behavior of such crowds may be modeled efficiently on a coarser level, treating its motion as the flow of a single aggregate system. By decomposing the scales of crowd motion in this way into an overall flow on the one hand and small deviations in individual motion on the other, I develop a novel inter-agent avoidance model which decouples the computational cost of local planning from the number of agents. The discrete motion of individuals is captured with a Lagrangian representation, while the macroscopic flow of the crowd is described with a coarser Eulerian model. This allows very large-scale crowds consisting of hundreds of thousands of agents to be simulated scalably at interactive rates.

In dense crowds, the finite spatial extent occupied by humans becomes a significant factor. This effect introduces new challenges, as the flow varies from freely compressible when the density is low to incompressible when the agents are close together. This characteristic is shared by many other dynamical systems consisting of numerous objects of finite size, including granular materials, hair, and dense traffic. I introduce in this chapter a mathematical formulation to model the dynamics of such aggregate systems in a principled way.

The key results presented in this chapter are:

- a hybrid representation for large crowds with discrete agents using both Lagrangian and Eulerian methods (section 4.3), and
- a continuum projection method that enforces density-dependent incompressibility to model the varying behavior of human crowds (section 4.4), resulting in

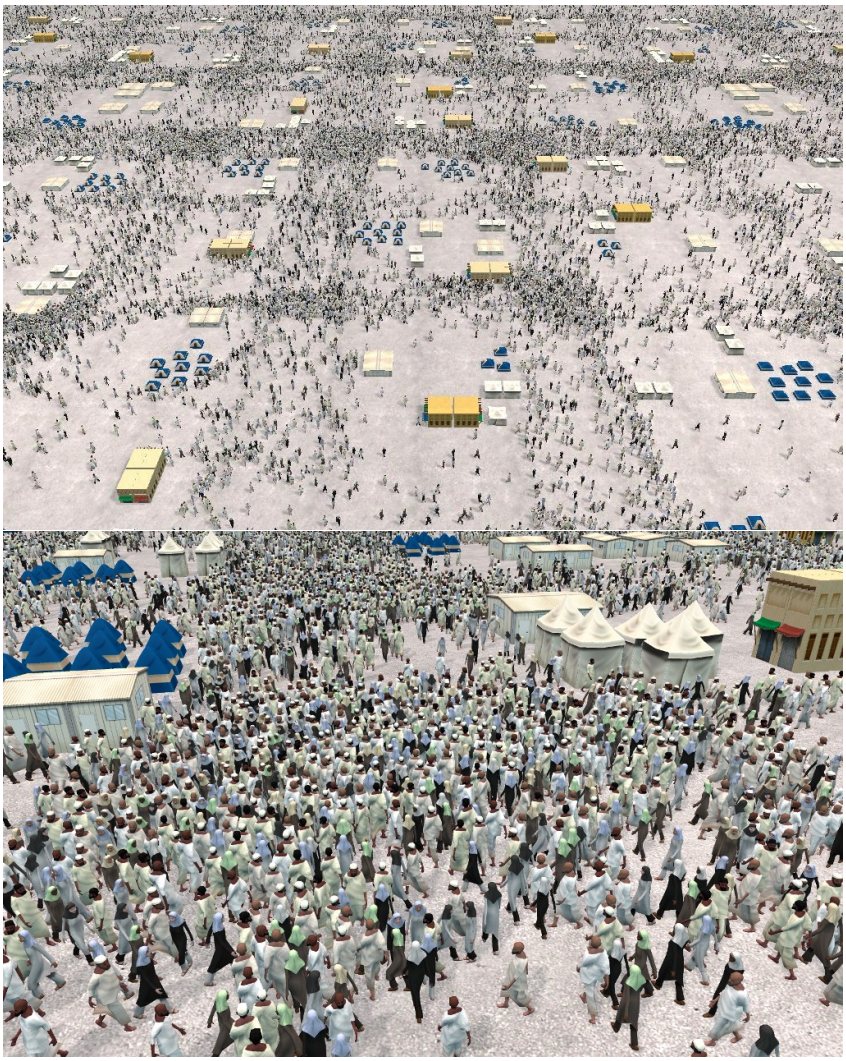


Figure 4.1: 100,000 pilgrims moving through a campsite, simulated with the scale decomposition technique.

- a scalable crowd simulation that can model hundreds of thousands of agents at interactive rates on current desktops (section 4.5).

The technique described here was previously published as Narain, Golas, Curtis, and Lin (2009). Figure 4.1 shows some of the results for dense crowds of up to 100,000 agents closely packed in complex scenes simulated at interactive rates with this technique. More analysis of the system's performance can be found in section 4.5.

4.2 RELATED WORK

Crowd simulation consists of many different components, including global planning, local avoidance, behavioral modeling, and motion synthesis. In the interest of space, I only touch upon the former two topics here, which are most directly related to my work. I refer the interested readers to the excellent surveys (Schreckenberg and Sharma, 2001; Thalmann et al., 2006; Pelechano et al., 2008; Pettré et al., 2008) for more details.

Many different techniques have been proposed for modeling the motion of multiple human agents in a crowd. The primary task in this problem is to compute each agent's path towards its goal, while avoiding collisions with obstacles and other agents and reproducing natural human behavior. On a broad level, the problem can be separated into *global* planning and *local* behavior.

Local models for agent behavior can be traced back to the seminal work of Reynolds (1987, 1999) who demonstrated emergent flocking and other behaviors from simple local rules. A large body of further work has accounted for sociological factors (Musse and Thalmann, 1997), psychological effects (Pelechano et al., 2005), directional preferences (Sung et al., 2004), social forces (Helbing and Molnár, 1995; Cordeiro et al., 2005; Gayle et al., 2009), and many other models of pedestrian behavior (Shao and Terzopoulos, 2007; Paris et al., 2007; Yeh et al., 2008). Many methods have also been proposed for collision avoidance between nearby agents. These include geometrically-based algorithms (Fiorini and Shiller, 1998; Feurtey, 2000; Sud et al., 2008; van den Berg et al., 2008; Guy et al., 2009; van den Berg et al., 2009), grid-based methods (Loscos et al., 2003; Lin et al., 2009), force-based methods (Heigeas et al., 2003; Lakoba et al., 2005; Sugiyama et al., 2001; Sud et al., 2007), Bayesian decision processes (Metoyer and Hodgins, 2003), and divergence-free flow tiles (Chenney, 2004).

Local control of agents cannot properly model the behavior of agents that aim to move towards specified goals, because of the possibility of getting stuck behind an obstacle. Therefore, local models are often combined with global planning techniques for practical

crowd simulation. These methods typically represent the collision-free space as a graph, and perform search to compute the path for each agent (Funge et al., 1999; Bayazit et al., 2002).

A continuum theory for the flow of pedestrians was proposed by Hughes (2003) and then extended by Treuille et al. (2006a). This approach combines global and local planning in a single optimization-based framework, and gives compelling results for many kinds of crowds. However, as Treuille et al. state, this approach is not suited for dense crowds where people are very closely packed and tend to come into contact.

4.3 DISCRETE AND CONTINUOUS CROWDS

The approach presented here is motivated by the observation that in a dense crowd, individual agents have a reduced freedom of movement, as they are tightly constrained by nearby agents. This observation has been exploited by previous work on continuum models for medium-density crowds (Hughes, 2003; Treuille et al., 2006a), and it has been noted that crowds at high density show behavior similar to granular flows (Helbing et al., 2005). Motivated by this observation, I present a crowd model that directly describes the aggregate motion of the crowd as a whole.

Specifically, in this approach, the standard representation of a crowd as a set of agents is augmented with a continuous representation, which characterizes the crowd as a continuum fluid described by a density and flow velocity. The problem of local collision avoidance is mapped into the continuous domain to obtain a novel variational constraint on the crowd flow, which I term the *unilateral incompressibility constraint* (UIC). This constraint acts as a large-scale collision avoidance step to accelerate the simulation.

This approach can be used as a local planning module in conjunction with a global planner, such as a roadmap-based algorithm or the continuum-based optimization (Treuille et al., 2006a) on a coarse grid, for the simulation of large dense crowds. The main simulation loop is structured as follows (figure 4.2):

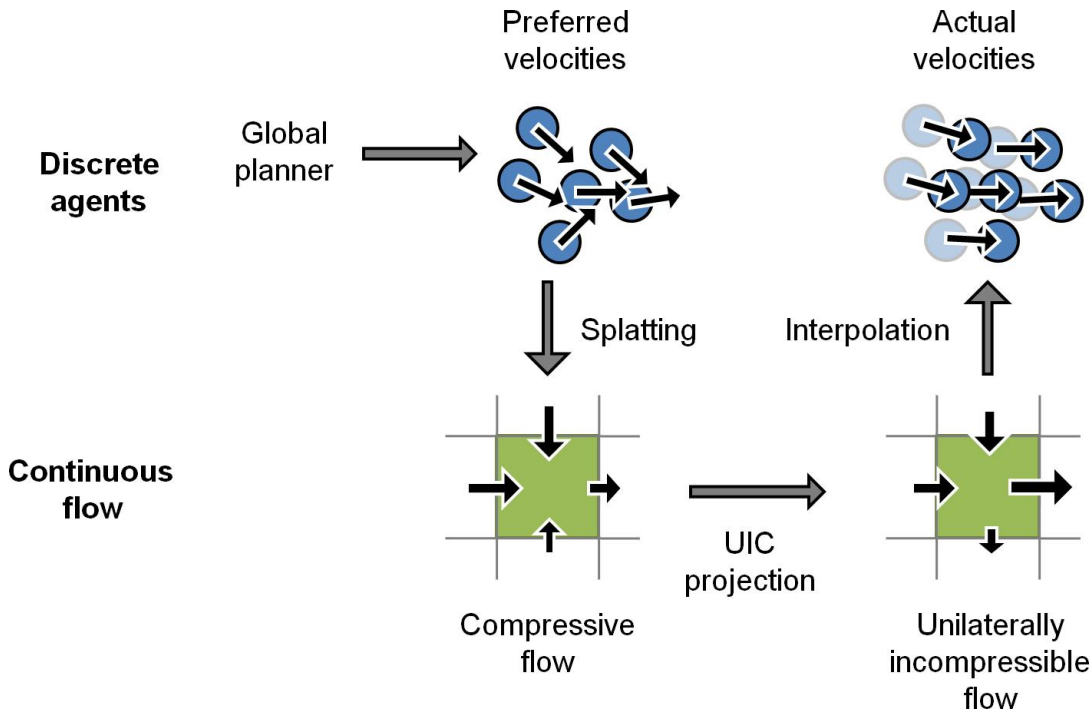


Figure 4.2: The system overview of the proposed crowd simulation algorithm.

1. At each timestep, each agent i determines its preferred velocities $\tilde{\mathbf{v}}_i$ using, for example, a global planner (section 4.3.1).

2. The discrete set of agents $\{\mathbf{x}_i, \tilde{\mathbf{v}}_i\}$ is converted to a continuous representation, yielding density ρ and velocity $\tilde{\mathbf{v}}$ (section 4.3.2).

3. In the continuous setting, the UIC is solved to performed inter-agent collision avoidance. This gives the corrected velocity field $\mathbf{v}$ (section 4.4).

4. Finally, agents determine their actual velocities $\mathbf{v}_i$ based on the velocities sampled from the continuous representation, and move forward according to these velocities (section 4.3.3).

The shift of perspective to a continuum approach has both conceptual and practical benefits. On the conceptual side, it represents the crowd in terms of macroscopic quantities which directly characterize the large-scale flow. In practical terms, this abstraction brings a significant performance benefit. Performing inter-agent collision avoidance for every

individual agent, as in existing methods, quickly becomes the most time-consuming step in crowd simulation, especially for dense crowds where the number of agents is large and the space of allowable collision-free motion is small. A continuum approach decouples this computational cost from the number of agents, allowing very large crowds to be simulated without being hindered by the inter-agent avoidance step.

In the remainder of this section, I set the background by describing the familiar discrete representation of a crowd and how it relates to the continuous model. The projection operation to solve the UIC in the continuous setting will then be described in section 4.4.

4.3.1 AGENT-LEVEL PLANNING

To determine the preferred velocities for each agent, a high-level model of agent behavior is necessary, such as a global planning step. Typically, global planning ignores the presence of other agents and determines a path that avoids the large static obstacles in the scene. In the larger crowd simulation framework, the global planner is used to determine the preferred velocities of agents, by taking the initial segment of the globally planned path as the preferred direction of motion. The task of reconciling these preferred velocities with collision avoidance among agents is then performed by the UIC solve.

In the implementation, a roadmap-based approach was used (see *e.g.* Bayazit et al. (2002); Sud et al. (2007, 2008) for details). However, the proposed technique is entirely agnostic to the particular global planner that is used; more complex behavioral models, a continuum planning approach, or user-scripted goal directions could be used instead. For completeness, the algorithm for roadmap-based global planning is outlined below.

Given a static scene with a set of polygonal obstacles, one first needs to define a roadmap which represents the connectivity of the scene as a graph. To construct the roadmap, the traversable scene regions are first tessellated using a constrained Delaunay triangulation T that avoids obstacles. Then the roadmap graph can be defined as follows.

The set of roadmap vertices V consists of the midpoints of all triangle edges that do not lie on an obstacle. Two roadmap vertices are connected if they lie on the same Delaunay triangle. Then, the shortest path between all pairs of vertices in V is precomputed using the Floyd-Warshall algorithm, with Euclidean distance being the edge weight metric.

To perform global planning for an agent, it is desired to find the shortest path through the scene between its current and goal positions. This is approximated by the shortest path between the roadmap vertices nearest these two points, which has already been precomputed. All other vertices on this path are considered sub-goals, and the preferred velocity of the agent is chosen to be in the direction of the visible sub-goal closest to the goal. Visibility is computed using a line sweep algorithm on the obstacle set O.

4.3.2 AGENTS AND THE GRID

The continuous fields ρ and $\mathbf{v}$ are stored on a staggered grid, as is common in fluid simulation. The density ρ is stored at both cell centers and edges, to facilitate the numerical method described in section 4.4. The x-component of velocity is stored on the cell edges normal to the x axis, and similarly for the y-component.

The transfer of information from discrete agents to the grid closely follows the particle-in-cell method of fluid simulation (Harlow, 1963; Zhu and Bridson, 2005), which was also used for crowd planning by Treuille et al. (2006a). The values of density ρ and velocity $\tilde{\mathbf{v}}$ on the grid are obtained by accumulating the values carried by nearby agents, weighted by the bilinear interpolation weights $w_{\mathbf{x}}(\mathbf{x}_i)$ associated with the agent position $\mathbf{x}_i$. That is,

$$\rho(\mathbf{x}) = \sum_i w_{\mathbf{x}}(\mathbf{x}_i) m_i, \tag{4.1}$$

$$\tilde{\mathbf{v}}(\mathbf{x}) = \frac{\sum_i w_{\mathbf{x}}(\mathbf{x}_i) \tilde{\mathbf{v}}_i}{\sum_i w_{\mathbf{x}}(\mathbf{x}_i)} \tag{4.2}$$

The mass m_i of each agent is taken to be unity. With this procedure, the crowd is converted to a continuous representation, on which the UIC projection can be performed.

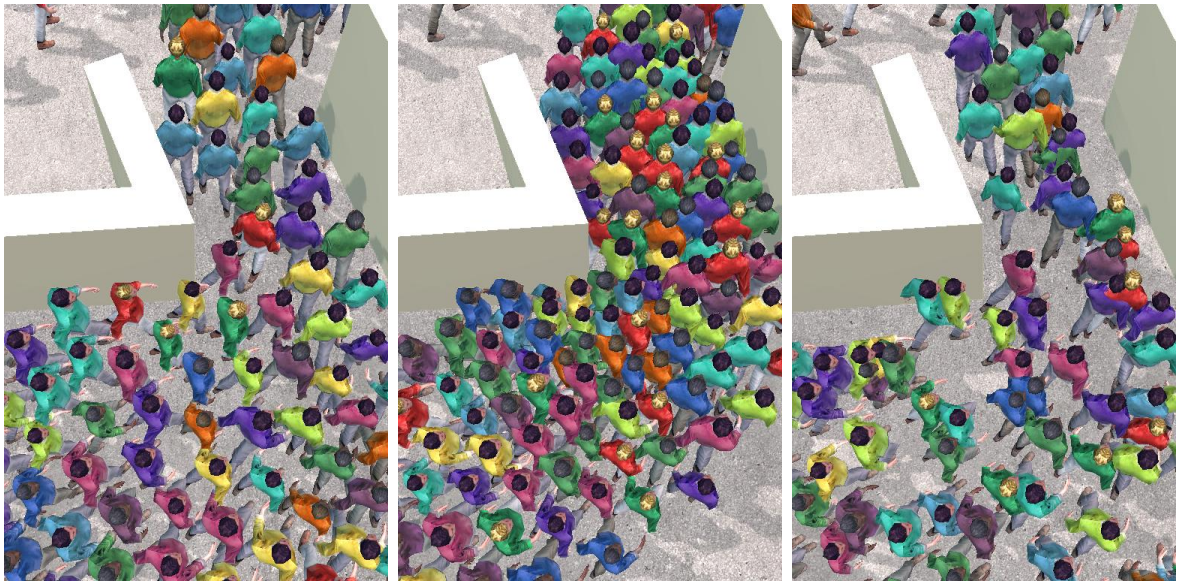


Figure 4.3: Agents attempting to exit through a narrow doorway, (a) with both the UIC projection and pairwise collision resolution, (b) without UIC, and (c) without collision resolution.

4.3.3 AGENT MOTION

The UIC projection, which will be described in detail in section 4.4, defines a flow field through ρ and $\mathbf{v}$ which prevents inter-agent collision on a macroscopic level. Here, I first describe how to use the computed flow field to determine the actual velocity of each agent and thus to execute its motion.

In dense regions, the agents are constrained by the flow of the crowd, and the interpolated grid velocity at the agent position $\mathbf{v}(\mathbf{x}_i)$ directly determines the velocity of agent i. However, in regions of low density, following the flow velocity completely would over-constrain the agents' motion. Accordingly, a weighted average of the continuum velocity $\mathbf{v}(\mathbf{x}_i)$ and the agent's own preferred velocity $\tilde{\mathbf{v}}_i$ is used, weighted by the crowd density $\rho(\mathbf{x}_i)$ at its location:

$$\mathbf{v}_i = \tilde{\mathbf{v}}_i + \frac{\rho(\mathbf{x}_i)}{\rho_{max}} \left(\mathbf{v}(\mathbf{x}_i) - \tilde{\mathbf{v}}_i\right). \tag{4.3}$$

Finally, the UIC projection can only act towards enforcing separation between agents on a gross level. It is still necessary to introduce an additional step to enforce minimum

distances for each pair of individual agents. This can be performed by simply moving agents apart if they are too close to each other, following Treuille et al. (2006a). While this simple pairwise collision resolution procedure is not guaranteed to separate all pairs exactly to the preferred distance, it gives good results because agent overcrowding is already prevented by the UIC. Figure 4.3 illustrates that both this step and the UIC solve are necessary to prevent inter-agent collisions.

4.4 THE UNILATERAL INCOMPRESSIBILITY CONSTRAINT

This section describes the operation that enforces volumetric constraints on the crowd, the primary contribution of this chapter. The role of this operation is a macroscopic counterpart to inter-agent collision avoidance. First, I will describe this in the mathematically continuous setting, then derive the numerical method that implements it on the simulation grid.

The crowd's state and desired motion are represented as continuous fields of density ρ and preferred velocity $\tilde{\mathbf{v}}$. Their evolution over time can be seen as a fluid-like system. However, it does not correspond directly to a physical fluid, as it is neither purely incompressible (a crowd can easily converge or disperse, changing its density), nor purely compressible (it is not possible to compress it indefinitely). Instead, I propose to treat a crowd as a new kind of "unilaterally incompressible" fluid, which is a hybrid of the two. This entails directly imposing an inequality constraint on ρ, preventing it from increasing beyond a maximum value $\rho_{\max}$ which corresponds to agents being packed as closely together as possible.

The method rests on the assumption that agents do not come closer together than a specified distance, say $d_{\min}$. This need not represent actual person-to-person contact, but simply a minimal comfortable distance to be maintained. From a macroscopic perspective,

this constraint implies an upper bound on the number density of agents,

$$\rho \leq \rho_{\max} = \frac{2\alpha}{\sqrt{3}d_{\min}^2}, \tag{4.4}$$

with $\rho_{\max}$ denoting the maximum allowed density. I call this the *unilateral incompressibility constraint* (UIC), following the terminology from mechanics of unilateral (inequality) vs. bilateral (equality) constraints. The constant factor $\alpha \leq 1$ is present to allow for the fact that perfect close packing is rarely achieved. This kind of constraint can in fact be applied not just to crowds, but to many other aggregate systems consisting of objects of a finite size, including granular materials and dense traffic flows.

Now, the actual flow, defined by $\mathbf{v}$, is chosen to be one which is close to the preferred flow $\tilde{\mathbf{v}}$, while maintaining the UIC (4.4). This idea is formalized in the following subsection.

4.4.1 UNILATERAL INCOMPRESSIBILITY

Intuitively, to prevent more agents from entering an already dense region, there should be a kind of positive pressure there which repels them. This does not necessarily correspond to a *physical* force; it is only a conceptual entity introduced to formulate the constraint in lieu of performing local planning for each individual. Accordingly, one can imagine introducing a scalar pressure field p which acts to enforce the constraint. The actual flow can then be defined as

$$\mathbf{v} = \tilde{\mathbf{v}} - \nabla p. \tag{4.5}$$

This is analogous to the way incompressibility is enforced in traditional fluid simulation. The only differences with pressure in physical fluids are that (1) where $\rho < \rho_{\max}$, p should be 0 since the constraint is not active, and (2) $p \geq 0$ everywhere to disallow negative pressure from pulling together agents that want to separate.

Formally, the problem is defined as follows. Consider the continuous model of the crowd as a constrained dynamical system, with an inequality-constrained state variable ρ which evolves under the action of the field $\tilde{\mathbf{v}}$. The relationship between ρ and $\mathbf{v}$ follows

from the fact that the number of agents is conserved:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{4.6}$$

To ensure that the UIC (4.4) is maintained at all times, a correction must be applied to $\tilde{\mathbf{v}}$ so that ρ remains within the feasible space. The issue thus is to determine the corrected velocity $\mathbf{v}$ which is in some sense "close" to $\tilde{\mathbf{v}}$ but maintains the UIC.

Assuming that agents will try to make as much progress in their desired direction as possible, the choice of $\mathbf{v}$ is determined by maximizing $\int \rho \mathbf{v} \cdot \tilde{\mathbf{v}}$, subject to some maximum walking speed, $||\mathbf{v}|| \leq v_{\max}$. In the implementation presented here, $v_{\max}$ is a constant, but it can also be made variable for each grid cell, depending on the density or the characteristics of nearby agents. For example, agents in dense regions have been observed to walk more slowly (Fruin, 1971), and this could be modeled by making $v_{\max}$ a function of ρ.

For the above condition, it can be shown via variational calculus that the optimal solution is of the form

$$\mathbf{v} = v_{\max} \frac{\tilde{\mathbf{v}} - \nabla p}{||\tilde{\mathbf{v}} - \nabla p||} \tag{4.7}$$

for some scalar "pressure" $p \geq 0$ satisfying

$$\rho < \rho_{\max} \Rightarrow p = 0, \tag{4.8}$$

$$p > 0 \Rightarrow \nabla \cdot \mathbf{v} = 0. \tag{4.9}$$

The interpretation of (4.7) is as follows. The crowd flow experiences a nonnegative pressure p which prevents the flow from converging to a density higher than $\rho_{\max}$. After the effect of pressure, agents amplify their adjusted velocity to attain the maximum allowed speed $v_{\max}$. Note that since (4.8) is also equivalent to

$$p > 0 \Rightarrow \rho = \rho_{\max}, \tag{4.10}$$

there is a *complementarity* between the unilaterally constrained terms ρ and p, in that either $\rho = \rho_{\max}$ or $p = 0$ must hold at any point. Intuitively, this means that pressure

only acts in regions when the density ρ is at the maximum. Where $\rho < \rho_{\max}$, agents are not densely packed and exert little influence on each other, and the crowd flows freely.

4.4.2 NUMERICAL METHOD

Here, I develop the numerical method for performing the UIC projection on the simulation grid.

Due to the nonlinearity of (4.7), it is not possible to solve it directly. Instead, it can be split into the composition of two operations,

$$\mathbf{v} = v_{\max}\frac{\tilde{\mathbf{v}} - \nabla p}{||\tilde{\mathbf{v}} - \nabla p||} = \mathsf{renorm}(\mathsf{psolve}(\tilde{\mathbf{v}})), \tag{4.11}$$

where $\mathsf{psolve}(\tilde{\mathbf{v}}) = \tilde{\mathbf{v}} - \nabla p$, and $\mathsf{renorm}(\hat{\mathbf{v}}) = v_{\max}\hat{\mathbf{v}}/||\hat{\mathbf{v}}||$. Note that collision avoidance is performed by psolve, while the role of renorm is merely to speed up the flow. To make analysis possible, let us neglect for now the effect of renormalizing velocities when solving for the pressure; this will be accounted for later.

Substituting $\mathbf{v} = \mathsf{psolve}(\tilde{\mathbf{v}})$ into the evolution equation yields

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho\tilde{\mathbf{v}}) + \nabla \cdot (\rho \nabla p). \tag{4.12}$$

In the discrete setting with cell size Δx, the gradient of pressure ∇p is defined at cell edges by finite differencing the pressure at adjacent cells. The divergence $\nabla \cdot \mathbf{u}$ of some vector field $\mathbf{u}$ on a cell is given by summing the flux over the cell edges,

$$\nabla \cdot \mathbf{u}\Delta x^2 \approx \sum_{\text{edges}} \mathbf{u} \cdot \mathbf{n}\Delta x. \tag{4.13}$$

At the nth timestep, the density values ρ^n and pre-projection velocities $\tilde{\mathbf{v}}^n$ are known. To first order, the density at the $(n+1)$th timestep is then given by

$$\rho^{n+1} = \rho^n - \nabla \cdot (\rho^n \tilde{\mathbf{v}}^n)\Delta t + \nabla \cdot (\rho^n \nabla p^n)\Delta t. \tag{4.14}$$

With discrete timesteps, it is preferable to apply UIC to the value at the next timestep, rather than instantaneously in differential form as in the previous subsection. Applying

the constraint (4.8) to ρ^{n+1} itself, a linear complementarity problem (LCP) is obtained. A general LCP is of the form

$$\mathbf{z} = \mathbf{A}\mathbf{x} + \mathbf{b},$$
$$\mathbf{z} \geq 0, \quad \mathbf{x} \geq 0, \quad \mathbf{z}^T\mathbf{x} = 0. \tag{4.15}$$

With the substitution $\sigma = \rho_{\max} - \rho$, it can be seen that the problem is equivalent to an LCP with

$$\mathbf{z} = \sigma^{n+1}, \tag{4.16}$$
$$\mathbf{b} = \sigma^n + \nabla \cdot (\rho^n \tilde{\mathbf{v}}^n)\Delta t, \tag{4.17}$$
$$\mathbf{A}\mathbf{x} = -\nabla \cdot (\rho^n \nabla \mathbf{x})\Delta t, \tag{4.18}$$
$$\mathbf{x} = p^n. \tag{4.19}$$

Also, because the matrix $\mathbf{A}$ is symmetric and positive semidefinite, the LCP is equivalent to a bound-constrained quadratic programming (QP) problem.

This problem has certain useful structure which can be exploited in order to solve it more efficiently than general-purpose QP or LCP solvers. In particular: it is symmetric, positive semidefinite and sparse; an efficient preconditioner, the MIC(0) preconditioner (Bridson and Müller-Fischer, 2007), is known; and only nonnegativity constraints exist on the variables. The algorithm of Dostál and Schöberl (2005) takes advantage of these properties, and can be used to solve the UIC projection efficiently. This is an iterative algorithm which can be warm-started with the pressure values from the previous timestep for improved performance.

Solving the LCP yields the pressure-corrected velocity, say $\hat{\mathbf{v}}$. However, because the effect of velocity renormalization was neglected in the pressure solve, $\mathsf{renorm}(\hat{\mathbf{v}})$ may no longer satisfy the pressure constraint (4.8). As this constraint is essential for collision avoidance, the pressure projection is applied again on the renormalized velocities again. The final velocity, thus, is

$$\mathbf{v} = \mathsf{psolve}(\mathsf{renorm}(\mathsf{psolve}(\tilde{\mathbf{v}}))). \tag{4.20}$$

Figure 4.4: UIC-driven agents (left) interacting with RVO-driven agents (right) in the same simulation.

In practice, numerical error can cause ρ^n to exceed $\rho_{\max}$ for one timestep. To prevent renorm from amplifying the transient corrective motion this causes, the inner psolve projection is performed with $\mathbf{b}$ clamped to nonnegative values.

4.4.3 OBSTACLES

If there are obstacles in the simulation domain, each grid cell that is partially or totally covered by an obstacle has a smaller area available for agents to be present. Therefore, it is simply the density constraint on that cell that has to be modified. For each grid cell, denote by f the fraction of area *not* covered by obstacles. This can of course be precomputed for static obstacles. Then, in the projection step, the maximum density is scaled by f to reflect the reduced available area. That is, one can simply use $\sigma = f\rho_{\max} - \rho$ and solve the LCP as usual. This simple procedure, inspired by Batty et al. (2007), allows obstacles to be handled without grid stairstepping.

With this approach, the multi-scale method can be mixed with other methods for agent-based planning (Helbing and Molnár, 1995; Reynolds, 1999; van den Berg et al., 2008) as well as scripted agents in the same simulation. For each time step, the individually controlled agents are advanced first, after which they are treated as obstacles in the continuum model so that the crowd motion will flow around them. In figure 4.4 this is demonstrated with a group of agents from the proposed model interacting with agents directed by the reciprocal velocity obstacle (RVO) method of van den Berg et al. (2008).

4.4.4 ALGORITHM SUMMARY

For clarity, here I summarize the full simulation loop.

1. At the beginning of each timestep, the position $\mathbf{x}_i$ of each agent are known.

2. Global planning is performed to determine the preferred velocity $\tilde{\mathbf{v}}_i$ of each agent, taking into account environmental obstacles but not neighboring agents. (For efficiency, the global planner may cache results from earlier timesteps, so that the full planning cost is not paid at each timestep.)

3. The agent positions $\mathbf{x}_i$ and preferred velocities $\tilde{\mathbf{v}}_i$ are transferred to the simulation grid by (4.1) and (4.2).

4. If there are moving obstacles in the environment, the free area f of each grid cell is recomputed.

5. The UIC solve is performed, giving the corrected velocity field $\mathbf{v} = \mathsf{psolve}(\mathsf{renorm}(\mathsf{psolve}(\tilde{\mathbf{v}})))$.

6. Each agent determines its actual velocity $\mathbf{v}_i$ taking the corrected flow into account via (4.3), and updates its position for the next timestep as $\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i \Delta t$.

7. Finally, pairwise collision resolution is performed on the new $\mathbf{x}_i$ to handle inter-agent collisions.

4.5 RESULTS AND DISCUSSION

This algorithm has been tested on several challenging scenes with large, dense crowds. These simulations show that the method can efficiently handle crowds of hundreds of thousands of agents without a notable increase in computation time.

Figures 4.5 and 4.6 show two examples that demonstrate the behavior of the method in some idealized scenarios. In the former, 10,000 agents attempting to cross over to the opposite points of a circle meet in the middle and form a vortex to resolve the deadlock. The latter example shows a four-way crossing of pedestrians under sparse, medium and dense flow. With sparse flow, streams of agents are able to pass through easily, but as the density increases, a complex pattern of lanes and vortices forms to resolve the congestion.

In figure 4.7 is shown a real-world scenario of 80,000 individual agents being evacuated from a trade show. Since there are many obstacles in the scene, evacuation causes congestion and a gradual outflow.

The Hajj pilgrimage is another real-life example of very large crowds. Figure 4.1 shows a scenario based on the Plain of Arafat campsite on the pilgrimage route. Through this large environment, 100,000 pilgrims travel in different directions and form lanes and vortices as they pass each other. The motion of pilgrims in al-Masjid al-Haram (the Grand Mosque) in Mecca can also be simulated. In figure 4.8, 25,000 agents with different goal directions form a dense, complex flow with no collisions.

4.5.1 PERFORMANCE

The performance of the algorithm was measured on an Intel Core i7-965 machine at 3.2 GHz with 5.99 GB of RAM. The timing results for all the examples are shown in Table 4.1. Even with very large numbers of agents, the method achieves close to interactive performance. It supports general scenarios with independent, heterogeneous agents; the number of unique goals has no effect on the performance.

Figure 4.10 shows a profile of the time spent in different stages of the algorithm, for

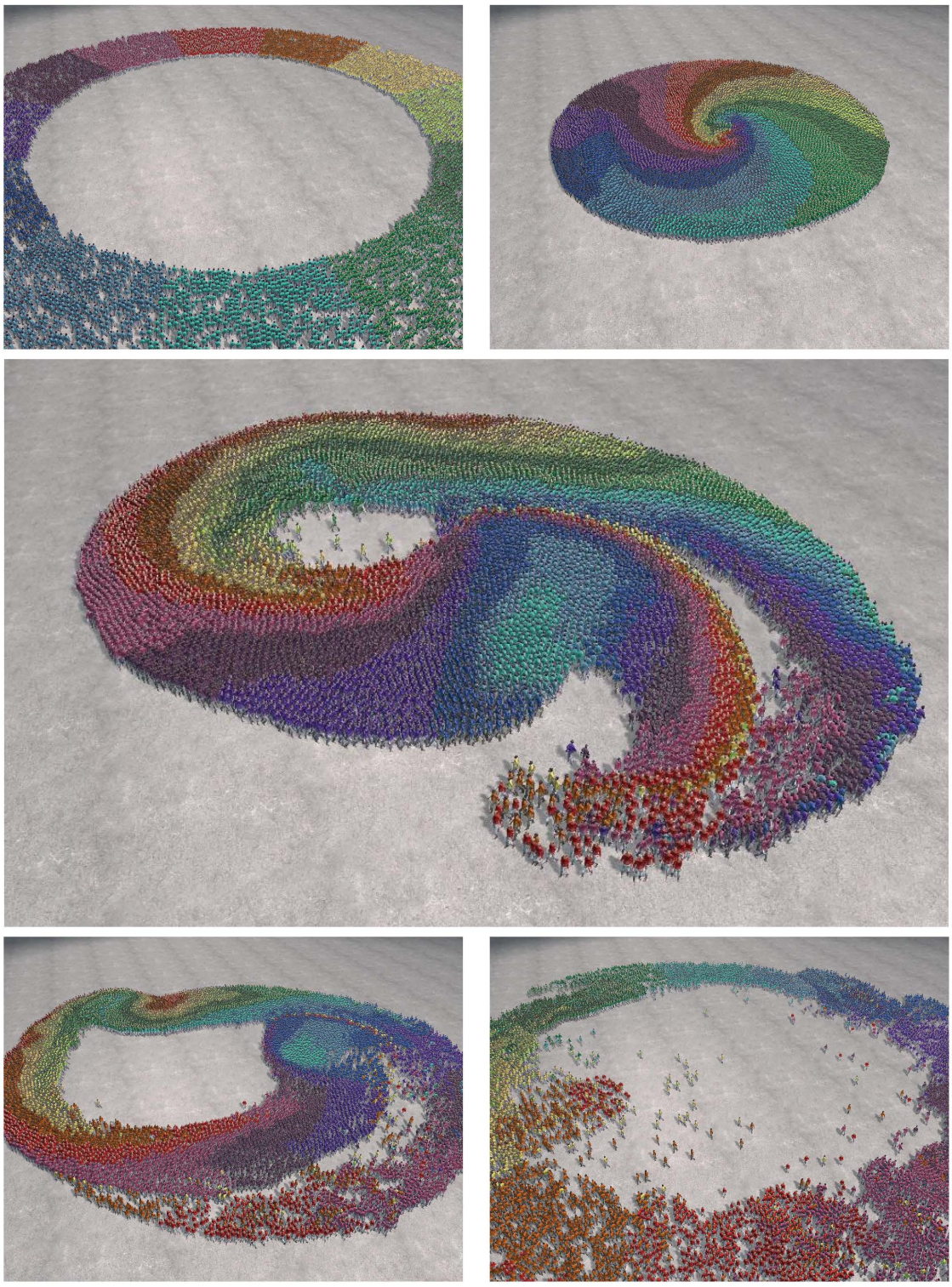


Figure 4.5: 10,000 agents in a circle moving to diametrically opposite points.

Figure 4.6: A four-way crossing of pedestrians entering the scene at rates of (a) 40, (b), 80, and (c) 120 agents per second.

Figure 4.7: 80,000 people evacuating a trade show.

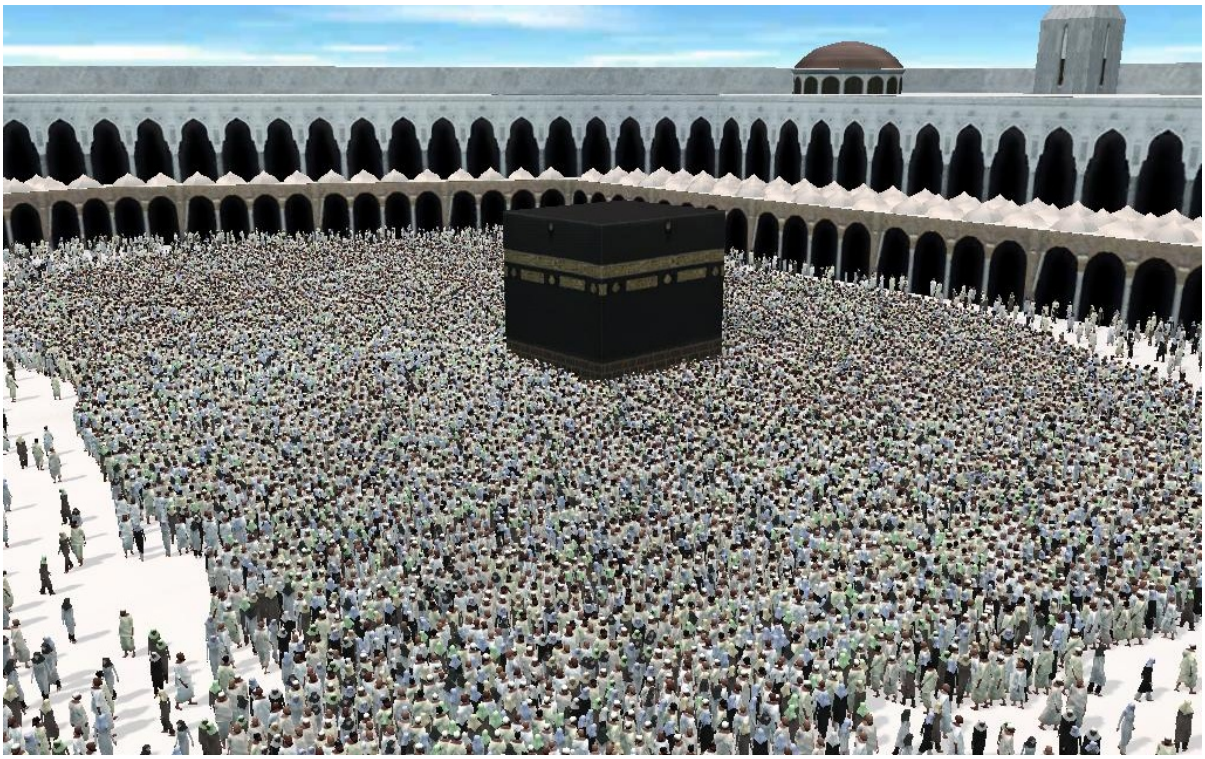


Figure 4.8: 25,000 pilgrims with heterogeneous goals in a mosque.

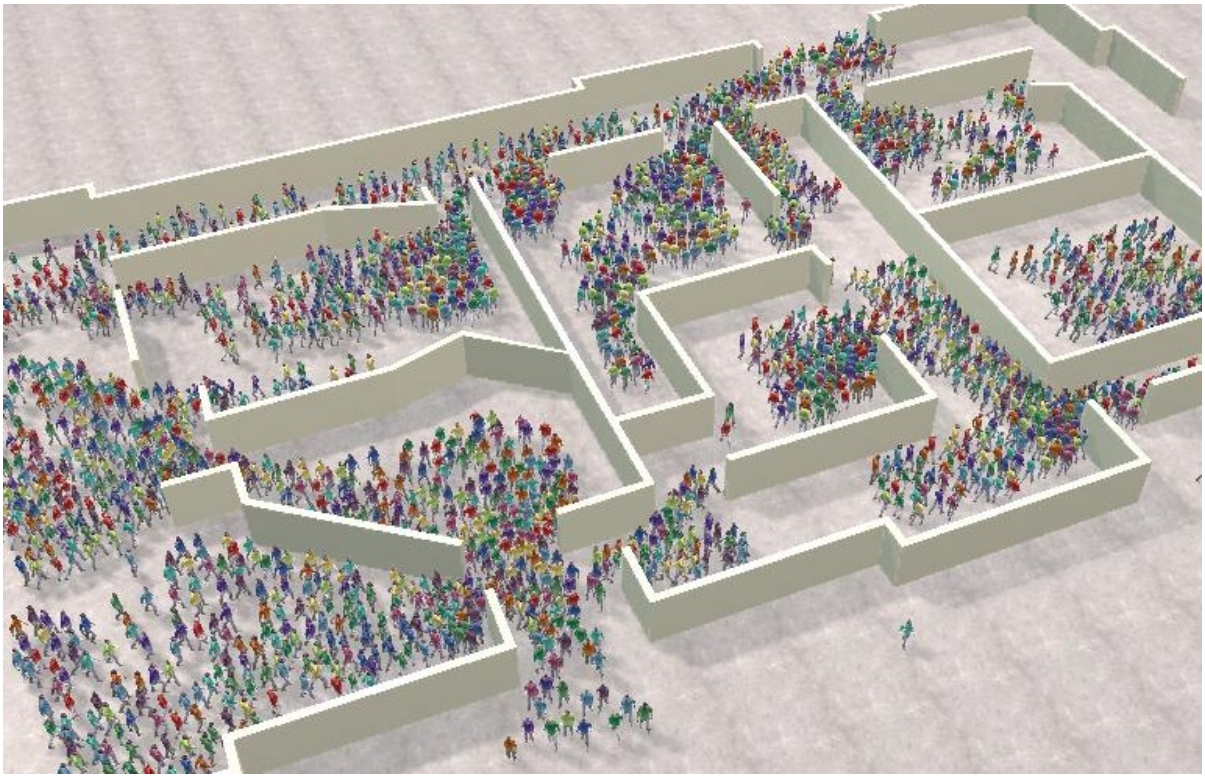


Figure 4.9: 2,000 agents evacuating a building.

Scenario	Agents	Grid size	Time/frame
Circle (Fig. 4.5)	10 k	40×40	34.2 ms
Crossing (Fig. 4.6)	6 k	40×40	16.5 ms
Building (Fig. 4.9)	2 k	90×60	16.1 ms
Trade show (Fig. 4.7)	80 k	120×108	806 ms
Campsite (Fig. 4.1)	100 k	120×90	447 ms
Mosque (Fig. 4.8)	25 k	80×80	88.1 ms

Table 4.1: The performance of the crowd simulation method on several scenarios.

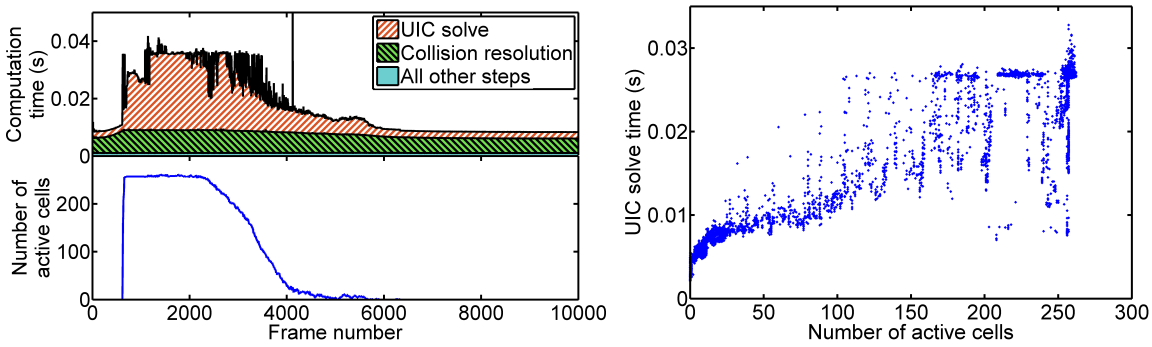


Figure 4.10: (a) The simulation time per frame (top) for the circle scenario, compared with the number of grid cells where the UIC constraint is active (bottom). The scenario contained 10,000 agents on a 40×40 grid. The majority of the simulation cost is in the collision resolution, which is largely constant, and the UIC solve, whose cost increases approximately linearly with the number of active cells (b).

the circle sequence of figure 4.5. The computational cost of the UIC solve is approximately linear with the number of actively constrained cells, *i.e.* those at maximum density, but is independent of the actual number of agents in the scene. The other expensive step is the pairwise collision resolution, which is an unavoidable per-agent cost.

One of the benefits of the UIC-based approach is that it allows a much simpler per-agent scheme to be employed. This is evident in figure 4.11, which shows how the proposed algorithm scales with the number of agents. For very large numbers of agents, the per-agent processing cost begins to dominate the computation time, but this cost is nevertheless much lower than that of existing methods. In particular, the method makes it possible to simulate a crowd of 1 million agents at 3 seconds per frame on a desktop computer. For comparison, the figure also shows the performance of the most

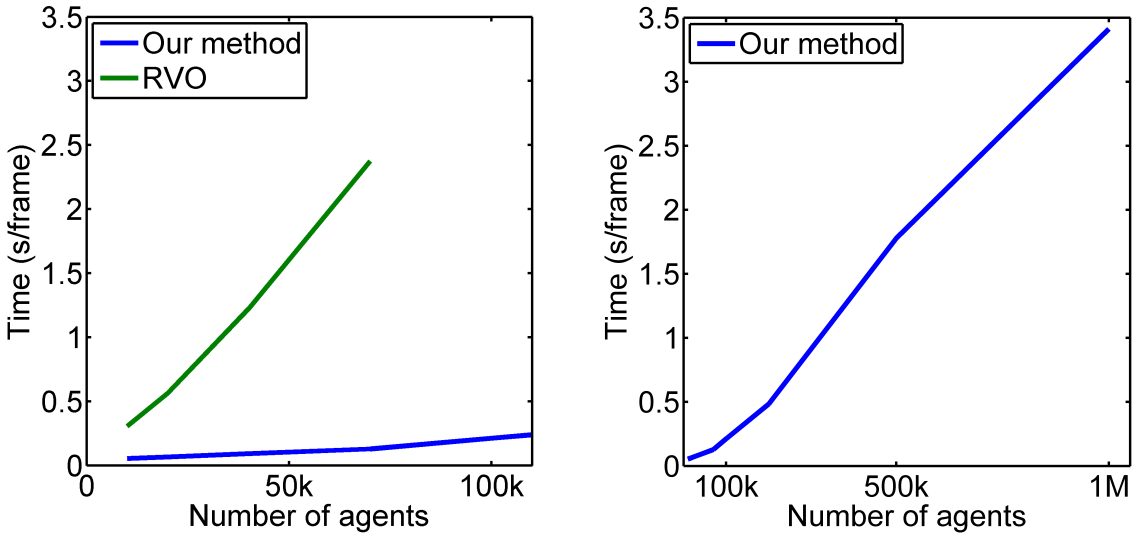


Figure 4.11: The proposed algorithm scales well with the number of agents up to 1 million, while the RVO method is more expensive and fails to run on simulations with more than 70,000 agents.

recent publicly-available implementation of the RVO method (van den Berg et al., 2008), a geometrically-based collision avoidance method. It failed to run on scenes containing more than 70,000 agents.

I also tested the effectiveness of this approach in performing pairwise collision avoidance. In experiments, moving colliding agents to a separation somewhat larger than the minimum intersection-free distance $d_{\min}$ after enforcing UIC gave smoother, jitter-free motion and a better spatial distribution of agents. For experimental validation, the distance of each agent to its nearest neighbor was measured over all frames of the circle scenario. The nearest distances have a mean of $1.25d_{\min}$ in the densest regions, and less than 0.12% of agents approach slightly closer than $d_{\min}$ over the entire sequence.

Over the different scenes shown in Table 4.1, the performance varies depending on the complexity of the scene, including static obstacles and the global planning roadmap. Also, when scaling up to a million agents, memory issues begin to be a significant factor. Overcoming these new bottlenecks is the next challenge that has to be addressed.

4.5.2 LIMITATIONS AND FUTURE WORK

Since this method is based on certain abstractions and approximations of the true individual-driven behavior of a crowd, it has some limitations when these approximations are not applicable.

Firstly, the pressure projection looks only at local information, and cannot anticipate future collisions from distant agents. Thus, two groups of agents approaching each other will not react until they are adjacent to each other. Since this requires non-local information, it may be better addressed as part of a global planning step instead. For example, the method could be used in conjunction with the potential fields approach of continuum crowds (Treuille et al., 2006a), which has global lookahead but does not handle the constraints of a dense crowd. In their original work, agent overlaps are prevented by pairwise collision resolution, which figure 4.3 shows to be inadequate for dense crowds, or by using a roughly agent-sized grid, which is expensive.

When defining the continuum constraint handling step, the current formulation optimizes only progress in the desired direction, in order to have a simple mathematical formulation. However, in real life human crowds display other behavioral features; for example, there is often an informal cultural convention to walk on one side of the path to avoid oncoming pedestrian traffic. It would be worthwhile to investigate the addition of such cultural and social behaviors to the method to enhance the realism of the results.

I anticipate that this approach can be combined with other techniques to enable a level-of-detail approach for simulating extremely large crowds. In specific regions where detailed individual motion is needed, such as close to the viewpoint, a more expensive agent-based scheme could be used, while distant agents would be efficiently simulated using the scale decomposition approach on a coarser grid. Adaptively refined grids can also be used for further control over the level of detail. This would achieve high quality behavior where it is desired, without sacrificing performance even for extremely large crowds.

Performing motion synthesis to map animated characters to the simulated paths is another challenge. In dense crowds, velocities fluctuate and often drop near zero, which leaves most simple approaches inadequate and vulnerable to foot skating artifacts. Generating more plausible character motion while remaining scalable to very large crowds is an open problem that I hope can be addressed by future work in motion synthesis.

CHAPTER 5

CONTINUUM SIMULATION OF GRANULAR MATERIALS

5.1 INTRODUCTION

Granular materials such as sand, powders, cereal grains, and gravel are commonplace in the physical world around us. Being composed of very large numbers of mesoscopic grains, they show unique physical behavior that is unlike other materials such as fluids and deformable bodies, which have been well studied in computer graphics. In particular, granular materials disperse freely in free fall, flow plastically under forcing, and yet settle in stable piles. The physical behavior of such materials arises from the interplay of contact and frictional forces between thousands to millions of tiny grains. Simulating the motion of each such grain is computationally prohibitive for large-scale scenarios or fine-grained materials like sand.

In this chapter, I present an efficient approach for simulating granular materials using scale decomposition, which combines a continuum model for granular flow with particle-based representations for advection and for rendering individual grains. The continuum model builds upon the formulation of unilateral incompressibility developed for crowd simulation in the previous chapter, and introduces additional techniques for handling the other forces and interactions within a granular material. This allows the fine-scale representation to be greatly simplified, while still tracking individual grains for realistic animation and rendering that preserves the granular appearance of the material.

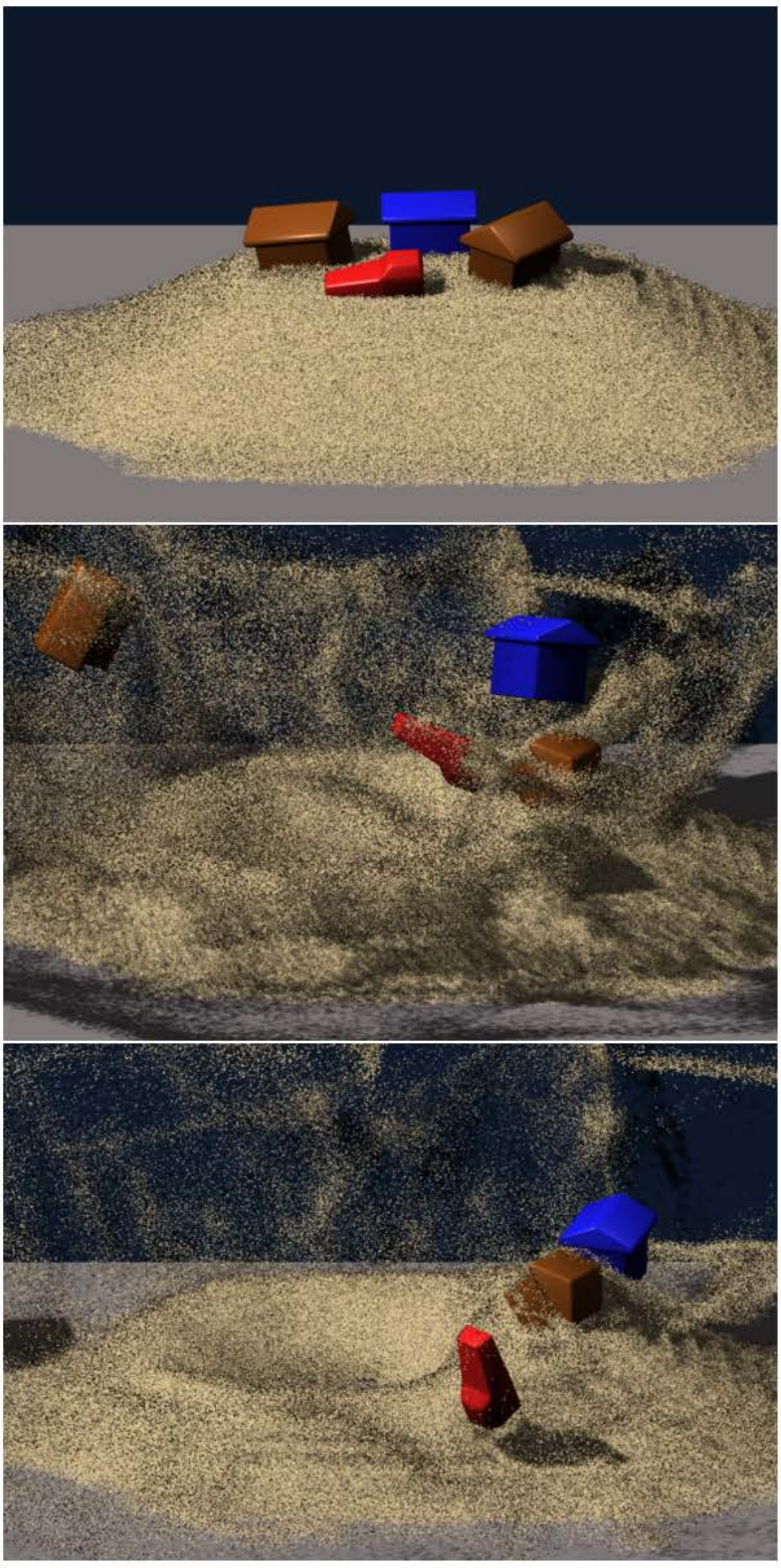


Figure 5.1: An explosion goes off inside a sand pile, sending freely splashing sand and rigid bodies flying in the air (running at less than 20 seconds per frame on a single-processor PC). In such a scenario, sand needs to be modeled as a cohesionless granular material.

In this approach, published previously as Narain, Golas, and Lin (2010), the continuum model is broadly similar to traditional fluids simulation techniques but departs significantly in technical content, in order to capture the unique behaviors that granular materials materials exhibit. Firstly, to allow the material to disperse freely when agitated but maintain its volume when at rest, as illustrated in figure 5.1, the existing fluid-based model's assumption of incompressibility is replaced with a unilateral incompressibility constraint. Secondly, unlike fluid viscosity, friction in granular materials can counteract gravity to maintain stable piles in equilibrium. This requires solving for the internal stresses in a global fashion. I present an efficient method for this numerical problem, permitting appropriate frictional behavior and solid body interaction.

5.2 RELATED WORK

Simulation of granular materials has been well studied in the fields of computational physics, and recently in computer graphics as well. Here I present a brief overview of related works, mainly in computer graphics, on this topic.

Physically-based simulation methods can be broadly divided into continuum-based and discrete approaches. Continuum-based approaches for granular materials abstract over individual grain motion, enabling efficient simulation of large volumes of material. Zhu and Bridson (2005) modeled sand as an fluid by adding frictional forces to a traditional fluid simulator. This method has been employed by Lenaerts and Dutré (2009) to simulate sand-water interaction. The incompressibility assumption, however, can lead to undesirable cohesive behavior. Continuum-based have also been employed in soil mechanics to study large-scale phenomena such as avalanches and landslides (Aranson and Tsimring, 2001; Quecedo et al., 2004; Josserand et al., 2004).

Discrete methods, on the other hand, directly simulate contact forces between individual grains. Luciani et al. (1995) developed a particle system model using damped spring forces. Bell et al. (2005) used a molecular dynamics method while modeling grains as rigid

compounds of spheres, exhibiting compelling behaviors at a significant computational cost. The computational issue was addressed in part by Alduán et al. (2009), who interpolate fine grains over the motion computed by a coarse discrete simulation.

Granular materials have been also modeled in computer graphics in other ways. Some of the earliest work used particle systems with heuristic inter-particle forces to give fluid-like or sand-like behavior (Miller and Pearce, 1989; Luciani et al., 1995). Sand has also been modeled using height fields (Li and Moshell, 1993; Chanclou et al., 1996), which was extended by Sumner et al. (1999) to model footprints and tracks. Onoue and Nishita (2003) used multi-valued height fields to achieve some 3D effects. Cellular automata were used by Pla-Castells et al. (2006) for interactive sand manipulation.

The method presented here builds on the continuum approach, and is more general, allowing non-cohesive behavior, globally coupled frictional handling, and efficient two-way interaction with solids. These characteristics allow qualitatively realistic, compelling results similar to fine-scale discrete methods at a far lower computation time and significantly reduced memory requirements.

5.2.1 OVERVIEW

In this method, it is assumed that the material's grain size is so small that the precise motion of individual grains is unimportant. Therefore, the granular material is treated instead as a continuous fluid flowing under the action of external forces and internal stresses. The internal stresses represent the contact and frictional forces between individual grains in the material. Computing the motion of the material under these forces requires two stages: first, to determine the internal stresses given the current state of the material, and second, to integrate the motion of the material under the influence of these forces.

While stress computation is most efficiently computed on a regular Eulerian grid, integrating the motion of a granular material proves to be best suited to a Lagrangian setting. The method is therefore based on a hybrid approach, informed by previous work

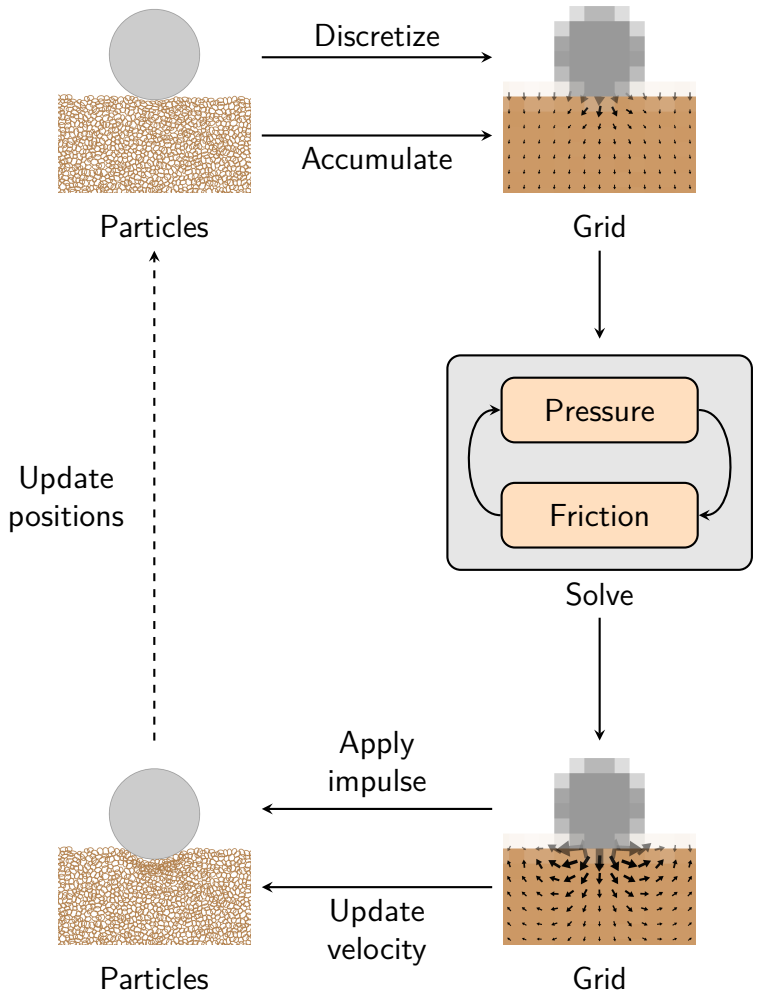


Figure 5.2: The main simulation loop of the algorithm. See section 5.2.1 for details.

such as the FLIP method (Zhu and Bridson, 2005), where an Eulerian representation is used to compute internal forces, while advection is performed using particles. Note that these simulation particles do not represent individual grains, but rather moving "clumps" of matter which act as samples of the material.

I briefly describe the overall simulation loop, illustrated in figure 5.2. At the beginning of a time step, the state of the material is projected to the Eulerian setting by accumulating particle values onto the grid. Using the Eulerian representation, the material's internal stresses and interaction forces with other bodies are computed by the continuum model, taking into account the coupling between pressure and friction (see figure 5.3). The

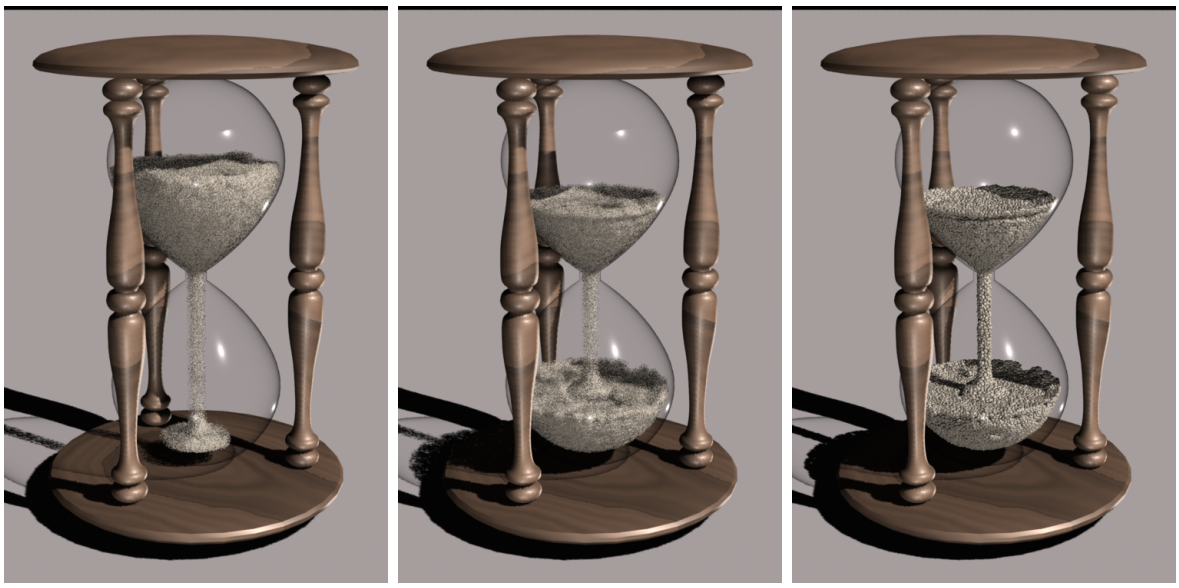


Figure 5.3: Sand trickles down an hourglass. The rate of flow is constant due to the coupling between pressure and friction. On the right, the simulation particles used for advection are visualized.

computed stresses are used by the Lagrangian representation to update the particle velocities. The motion of each particle is then integrated over the time step, giving the updated state of the material.

The remainder of the paper is organized as follows. The continuum model is described in section 5.3, and the particle-based advection scheme in section 5.4. Further implementation details are given in section 5.5. In section 5.6, I discuss the results of the method on several example scenarios.

5.3 GRANULAR MATERIAL AS A CONTINUUM

The continuum-based approach has much in common with traditional fluid simulation. However, the behavior of stresses in a granular material differs qualitatively from pressure and viscosity in a fluid, and novel techniques are required to treat them faithfully in a continuum model.

One characteristic property of granular materials is that they can disperse freely. Dry grains apply no attractive forces on each other, so in macroscopic terms the material

displays little cohesion. This means that unlike a liquid, a granular material in motion may not have a clearly defined surface at all! Consider a sand pile transitioning to a cloud of grains under impact: where does the pile surface end and the cloud begin? Another significant difference is the effect of friction. Friction can transmit forces across large distances, and unlike fluid viscosity, it continues to act even at rest, resulting in stable piles and other quasi-rigid behaviors.

Indeed, many of the problems encountered in this work mirror those of rigid body dynamics transferred to a continuum setting, which is not surprising since granular materials are essentially aggregates of numerous rigid bodies in contact. In order to capture their complex behavior, ideas from both the fields of computational fluid dynamics and rigid-body simulation are combined and built upon to derive the new formulation.

5.3.1 FUNDAMENTALS

From a continuum viewpoint, one can represent the physical state of a granular material through its mass density ρ and flow velocity $\mathbf{v}$. The system is acted upon by external forces $\mathbf{f}_{\text{ext}}$ and internal stress σ. The stress σ is a symmetric tensor field which can be decomposed into an isotropic mean stress (a "pressure") p and a traceless deviatoric component $\mathbf{s}$, which represents frictional stresses.

$$\sigma = -p\mathbf{I} + \mathbf{s}. \tag{5.1}$$

The equations of motion of the material may be derived from the conservation laws for mass and momentum. If we denote the Lagrangian time derivative by $\mathrm{D}/\mathrm{D}t = \partial/\partial t + (\mathbf{v}\cdot\nabla)$, the time evolution of the system is given by the transport of mass,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{5.2}$$

and the effect of forces,

$$\rho \frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} = \mathbf{f}_{\text{ext}} - \nabla p + \nabla \cdot \mathbf{s}. \tag{5.3}$$

The stress response of frictional materials can be described in terms of plastic yielding. The material resists deformation when the stress is within a certain *yield criterion*, and begins to flow plastically when the yield criterion is violated. Following the previous work of Zhu and Bridson (2005), the Drucker-Prager yield criterion is employed here for its computational simplicity. When the cohesion is negligible, as is the case for most dry granular materials, the yield criterion can be expressed by the inequality

$$\|\mathbf{s}\|_F \leq \sqrt{3}\alpha p \tag{5.4}$$

where $\|\mathbf{s}\|_F = \sqrt{\sum s_{ij}^2}$ is the Frobenius norm of $\mathbf{s}$, and α is the frictional coefficient. This is related to the angle of repose θ by the relationship $\alpha = \sqrt{2/3}\sin\theta$.

The simulation uses an impulse-based time-stepping scheme, where forces are considered to act instantaneously at the beginning of each time step. Let the density and velocity field of the material at the beginning of time step n be ρ^n and $\mathbf{v}^n$ respectively. For a time step of length Δt, the intermediate velocity field $\tilde{\mathbf{v}}$ for given stress components p and $\mathbf{s}$ may be defined as

$$\tilde{\mathbf{v}} = \mathbf{v}^n + \frac{\Delta t}{\rho^n}\left(\mathbf{f}_{\text{ext}} - \nabla p + \nabla \cdot \mathbf{s}\right). \tag{5.5}$$

This velocity field may be used to update the particle velocities and positions in the advection step.

5.3.2 PRESSURE

The model rests on two simplifying assumptions regarding the dynamics of granular materials. Firstly, it is assumed that the material cannot be compressed beyond a fixed *critical density*, $\rho_{\max}$, which permits free flow. This can be thought of as the density of a stable pile of material at rest. (In reality, this density is slightly higher than critical, as grains at rest must separate a little to move past each other, but this effect is visually imperceptible.) Secondly, when $\rho < \rho_{\max}$, grains are not packed together and only interact via intermittent collisions, and the effect of these interactions is neglected.

Under these assumptions, the granular material has a maximum density, $\rho_{\max}$, and the pressure p acts to prevent any further compression of the material. However, in contrast to traditional incompressible fluids, where both positive and negative velocity divergence are always nullified by the pressure, an absence of cohesion in granular material implies that a material undergoing diverging flow experiences no internal forces.

This behavior can be expressed as an inequality on the density of the material, or more generally on its "volume fraction" $\phi = \rho/\rho_{\max}$, such that

$$\phi = \frac{\rho}{\rho_{\max}} \leq 1. \tag{5.6}$$

This is essentially the unilateral incompressibility constraint, which was applied in two dimensions to simulation of dense crowds of pedestrians in the previous chapter. Here, the same numerical method is adapted for computing the corresponding pressure p to satisfy the constraint in 3D space.

The volume fraction at the end of a time step Δt can be estimated by discretizing (5.2) with $\mathbf{v} = \tilde{\mathbf{v}}$. This gives

$$\phi^{n+1} = \phi^n - \Delta t \nabla \cdot (\phi^n \tilde{\mathbf{v}}) \tag{5.7}$$

$$= \phi^{n+1}|_{p=0} + \frac{\Delta t^2}{\rho_{\max}} \nabla^2 p, \tag{5.8}$$

where $\phi^{n+1}|_{p=0}$ is the predicted volume fraction when pressure is zero.

The pressure p is chosen so that the constraint (5.6) is satisfied for ϕ^{n+1}. Assuming inelasticity, the pressure must be such that it only just maintains the constraint, but does not cause any additional "bounce". This implies a complementarity

$$p(1 - \phi^{n+1}) = 0 \tag{5.9}$$

That is, when p is active (nonzero), ϕ^{n+1} must equal 1, not fall below it. Finally, the absence of cohesion implies that $p \geq 0$, which can also be seen from the yield condition (5.4) itself.

These conditions define a linear complementarity problem

$$\begin{aligned} \mathsf{A}_1\mathsf{p} + \mathsf{b}_1 &\geq 0, \\ \mathsf{p} &\geq 0, \\ \mathsf{p}^T(\mathsf{A}_1\mathsf{p} + \mathsf{b}_1) &= 0, \end{aligned}$$

where

$$\mathsf{A}_1 = \frac{\Delta t^2}{\rho_{\max}}\mathsf{D}_1^T\mathsf{D}_1, \tag{5.10}$$

$$\mathsf{b}_1 = 1 - \phi^{n+1}|_{p=0}, \tag{5.11}$$

and p is a vector containing the pressure values at all 3D grid cells. D_1 denotes the finite difference matrix mapping a scalar field p to the vector field ∇p. Boundary conditions are treated using the standard technique of ghost cells, on which $\phi^{n+1}|_{p=0}$ is set to 1 to prevent any flow across domain boundaries.

Because A_1 is positive semidefinite, these are simply the KKT conditions for minimizing a quadratic function

$$F = \frac{1}{2}\mathsf{p}^T\mathsf{A}_1\mathsf{p} + \mathsf{b}_1^T\mathsf{p}, \tag{5.12}$$

for p with all non-negative components. Therefore, the pressure p is equivalently defined by the minimization problem

$$\min\ F(\mathsf{p}) : \mathsf{p} \geq 0. \tag{5.13}$$

Density correction: Numerical error in advection can cause the density to violate (5.6) slightly at the beginning of a time step, which would cause the solver to apply spurious corrective pressure at those cells and lead to oscillations. To avoid this issue, it is necessary to first redistribute the density so the initial state is valid.

Imagine freezing the material in time and applying an instantaneous displacement $\Delta\mathbf{x}$ such that each bit of material at $\mathbf{x}$ is moved to $\mathbf{x} + \Delta\mathbf{x}$. One can choose $\Delta\mathbf{x}$ such that

Figure 5.4: Three stable piles with friction coefficients $\alpha = 0.3$, 0.5, and 0.7, respectively. Lower friction leads to flatter piles.

the new volume fraction

$$\phi^n \leftarrow \phi^n - \nabla \cdot (\phi^n \Delta \mathbf{x}) \tag{5.14}$$

satisfies (5.6). Letting $\Delta \mathbf{x} = -\nabla y / \rho^n$ for some scalar field y, this can be solved using the pressure solve itself, substituting $\Delta \mathbf{x}$ for $\tilde{\mathbf{v}}$ and 1 for Δt.

At the beginning of the time step, ϕ^n is updated with (5.14), and ρ^n is updated correspondingly. The displacement $\Delta \mathbf{x}$ is also stored for use during advection.

5.3.3 FRICTION

The frictional stress $\mathbf{s}$ is a symmetric, trace-free, rank-2 tensor,

$$\mathbf{s} = \begin{bmatrix} s_{xx} & s_{xy} & s_{xz} \\ s_{xy} & s_{yy} & s_{yz} \\ s_{xz} & s_{yz} & s_{zz} \end{bmatrix},$$

$$s_{xx} + s_{yy} + s_{zz} = 0,$$

subject to the yield condition (5.4). Because the yield constraint is convex, it can be linearized by replacing the constraint surface with a set of hyperplanes tangent to it. Taking the hyperplanes orthogonal to each component of $\mathbf{s}$ yields bound constraints

$$-s_{\max} \leq s_{ij} \leq s_{\max} \tag{5.15}$$

for $i, j \in \{x, y, z\}$, and with $s_{\max} = \alpha p$. If desired, more hyperplanes can be included to improve the isotropy of the frictional response. The effect of different values of α is shown in figure 5.4.

For plastic flow, the principle of maximum plastic dissipation (Simo and Hughes, 1998) states that among all possible stresses satisfying the yield criterion, the actual stress is that which maximizes the rate of dissipation of kinetic energy. The frictional stress is computed by directly applying the maximum dissipation principle over discrete time steps. This automatically captures the interplay between pressure and friction within the material both in motion and at rest, eliminates the need for an additional rigidity condition for stable piles, and naturally generalizes to interaction with solid bodies.

To maximize dissipation, one must find the frictional stress that minimize the kinetic energy of the system. However, from a numerical perspective, directly using the total kinetic energy results in a poorly conditioned system due to the division by ρ^n in the definition of $\tilde{\mathbf{v}}$. This can be ameliorated by using an additional weighting $w = \rho^n/\rho_{\max}$ on the energy:

$$E = \frac{1}{2}\int w\rho^n \left\|\tilde{\mathbf{v}}\right\|^2 \mathrm{d}V \tag{5.16}$$

This modification greatly improves the conditioning of the problem. Furthermore, because p and hence $\mathbf{s}$ are only ever nonzero when ρ^n is close to $\rho_{\max}$, w is 1 over almost the entire support of $\mathbf{s}$ so only a small amount of error is introduced.

The energy can be expressed as a quadratic functional

$$\begin{aligned} E &= \frac{1}{2\rho_{\max}}\int \left\|\rho^n\tilde{\mathbf{v}}|_{\mathbf{s}=0} + \Delta t\nabla\cdot\mathbf{s}\right\|^2 \mathrm{d}V \\ &= E|_{\mathbf{s}=0} + \frac{1}{2}\mathsf{s}^T\mathsf{A}_2\mathsf{s} + \mathsf{b}_2^T\mathsf{s}. \end{aligned} \tag{5.17}$$

where

$$\mathsf{A}_2 = \frac{\Delta t^2}{\rho_{\max}}\mathsf{D}_2^T\mathsf{D}_2, \tag{5.18}$$

$$\mathsf{b}_2 = \frac{\Delta t}{\rho_{\max}}\mathsf{D}_2^T\rho^n\tilde{\mathbf{v}}|_{\mathbf{s}=0}. \tag{5.19}$$

Here s is treated as the vector composed of the concatenation of the components of frictional stress at all grid points, and D_2 is the matrix mapping a tensor field $\mathbf{s}$ to the

vector field $\nabla \cdot \mathbf{s}$. Minimizing E subject to the constraints (5.15) determines the frictional stress:

$$\min\ E(\mathsf{s}) : -s_{\max} \leq s_{ij} \leq s_{\max}. \tag{5.20}$$

Boundary conditions are treated as follows. Physically, the normal force at the boundary due to friction must vanish. This corresponds to the diagonal components s_{ii} on the ghost cells, so these are fixed at zero. The tangential force corresponds to the off-diagonal components on edges along the boundary; the associated finite difference stencil does not refer to cells outside the domain at all, so no boundary conditions need to be specified here.

5.3.4 INTERACTION WITH SOLID BODIES

By posing both the pressure and friction solves as minimization problems, two-way solid-fluid interaction (as shown in figure 5.5) can be very naturally handled along the lines of variational fluid-solid coupling as described by Batty et al. (2007).

For a solid body interacting with a fluid or granular material, one can define a linear operator J which integrates stresses on its surface to give generalized forces. For example, for rigid bodies, generalized forces can be represented as 6-vectors with 3 components of force followed by 3 components of moment. The J operator for scalars then is given by

$$\mathsf{J}_1 p = \int \begin{bmatrix} \nabla p \\ \nabla p \times \mathbf{x} \end{bmatrix} \phi_{\mathrm{s}} \mathrm{d}V, \tag{5.21}$$

while that for tensorial stresses is

$$\mathsf{J}_2 \mathbf{s} = \int \begin{bmatrix} \nabla \cdot \mathbf{s} \\ (\nabla \cdot \mathbf{s}) \times \mathbf{x} \end{bmatrix} \phi_{\mathrm{s}} \mathrm{d}V, \tag{5.22}$$

in terms of the fraction of volume ϕ_{s} covered by the solid body. Under given material stresses p and $\mathbf{s}$ and external forces $\mathbf{F}_{\mathrm{ext}}$, the net generalized force on the body is simply

$$\mathbf{F} = \mathbf{F}_{\mathrm{ext}} - \mathsf{J}_1 \mathsf{p} + \mathsf{J}_2 \mathsf{s}. \tag{5.23}$$

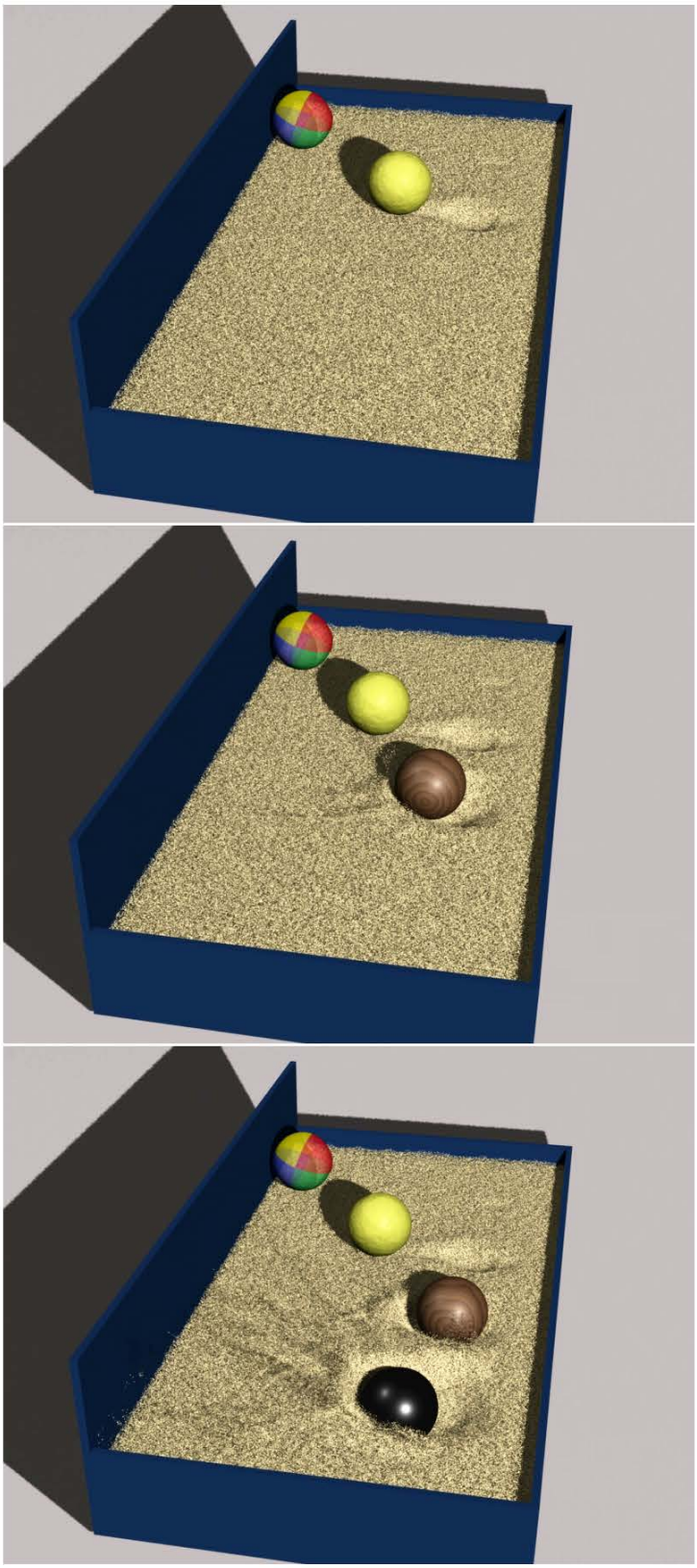


Figure 5.5: Spheres of masses $m = 0.3, 1, 3$, and 10 units strike a sand surface. The heaviest spheres are denser than the sand itself, yet internal friction allows the material to support their weight.

Furthermore, the adjoint J_1^T yields a scalar field $\mathsf{J}_1^T\mathbf{V}$ which gives the distribution of normal velocity times differential area over the surface of the body; intuitively, this describes how the space occupied by the body changes due to its velocity.

Consider a body with inertia matrix $\mathbf{M}$ and initial generalized velocity $\mathbf{V}^n$. After applying the impulse $\Delta t\mathbf{F}$, it moves with the updated velocity $\tilde{\mathbf{V}} = \mathbf{V}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}$.

For the pressure coupling, the volume fraction constraint (5.6) is replaced with

$$\phi + \phi_\mathrm{s} \leq 1. \tag{5.24}$$

Enforcing this constraint requires estimating the volume fraction covered by the body at the next time step. Using the adjoint property of J_1, this is given by

$$\begin{aligned}\phi_\mathrm{s}^{n+1} &= \phi_\mathrm{s}^n + \Delta t\mathsf{J}_1^T\tilde{\mathbf{V}} \\ &= \phi_\mathrm{s}^{n+1}|_{p=0} - \Delta t^2\mathsf{J}_1^T\mathbf{M}^{-1}\mathsf{J}_1\mathsf{p},\end{aligned} \tag{5.25}$$

where $\phi_\mathrm{s}^{n+1}|_{p=0}$ is the predicted volume fraction without pressure coupling. Thus, the interaction term $\Delta t^2\mathsf{J}_1^T\mathbf{M}^{-1}\mathsf{J}_1$ is added to the matrix A_1, and $-\phi_\mathrm{s}^{n+1}|_{p=0}$ to the linear term b_1 in (5.10) and (5.11) respectively.

In the friction solve, the only change needed is to add the kinetic energy of the rigid body to E. This is given by

$$\begin{aligned}E_\mathrm{s} &= \tilde{\mathbf{V}}^T\mathbf{M}\tilde{\mathbf{V}} \\ &= E_\mathrm{s}|_{\mathbf{s}=0} + \frac{1}{2}\Delta t^2\mathsf{s}^T\mathsf{J}_2^T\mathbf{M}^{-1}\mathsf{J}_2\mathsf{s} + \Delta t\mathsf{s}^T\mathsf{J}_2^T\tilde{\mathbf{V}}|_{\mathbf{s}=0}.\end{aligned} \tag{5.26}$$

Thus, $\Delta t^2\mathsf{J}_2^T\mathbf{M}^{-1}\mathsf{J}_2$ is added to the matrix A_2 and $\Delta t\mathsf{J}_2^T\tilde{\mathbf{V}}|_{\mathbf{s}=0}$ to the linear term b_2 in (5.18) and (5.19).

It is also necessary to modify the velocity updates of the granular material to account for the reduced volume occupied by it. Assuming that the body forces are distributed volumetrically in a partially occupied cell, the velocity update rule (5.5) becomes

$$\tilde{\mathbf{v}} = \mathbf{v}^n + \frac{\Delta t}{\rho^n}(1 - \phi_\mathrm{s})\left(\mathbf{f}_\mathrm{ext} - \nabla p + \nabla\cdot\mathbf{s}\right). \tag{5.27}$$

Once the pressure and frictional stresses are computed, the rigid body can be updated by applying an impulse $\Delta t\mathbf{F}$ using (5.23) and advancing it through one time step.

5.3.5 PUTTING IT TOGETHER

Given the current state of the system in terms of density ρ and velocity $\mathbf{v}$, the pressure p and frictional stress $\mathbf{s}$ are determined by minimization of two coupled quadratic programs $F(p)$ and $E(\mathbf{s})$, subject to corresponding linear inequality constraints. The problem of solving these coupled minimizations mirrors that addressed by Kaufman et al. (2008), who treated the contact and frictional forces between rigid bodies in the same way. This solution procedure, using staggered projections on each minimization in turn, extends to the continuous case considered here.

The staggered projection method works by fixing the value of one variable, say $\mathbf{s}$, and finding p through (5.13) using the current value of $\mathbf{s}$. Then, the resulting p is fixed and $\mathbf{s}$ is updated through (5.20). This pair of minimizations forms one iteration, which is repeated. This procedure of solving (5.13) and (5.20) can be shown to be a non-expansive mapping, of which the coupled solution is a fixed point. In the discretized setting, both problems become quadratic programs which can be solved efficiently, as will be described in section 5.5. Because of the non-expansive property of staggered projection iterations, the iteration process is guaranteed not to diverge, and in practice it is observed to converge extremely quickly.

More explicitly, the pressure p and frictional stress $\mathbf{s}$ are defined by a pair of coupled quadratic programs,

$$\min F(p) : p \geq 0, \tag{5.28}$$

$$\min E(\mathbf{s}) : -s_{\max} \leq s_{ij} \leq s_{\max}. \tag{5.29}$$

Below, I show that applying staggered projections has the simultaneous solution of both these minimizations as a fixed point. This proof closely follows section 6 of Kaufman et al. (2008), but works in the continuous setting.

The above minimizations can be reinterpreted as projections in impulse space. To clarify the discussion, let us introduce symbols to denote the predicted momentum of the material, and the impulses imparted by pressure and friction:

$$\mu_0 = \rho^n \left(\mathbf{v}^n + \frac{\Delta t}{\rho^n} \mathbf{f}_{\text{ext}} \right), \tag{5.30}$$

$$\mu_p = -\Delta t \nabla p, \tag{5.31}$$

$$\mu_s = \Delta t \nabla \cdot \mathbf{s}. \tag{5.32}$$

The projection of some impulse μ to its nearest point in a convex set A is defined as

$$P(\mu; A) = \underset{\nu \in A}{\operatorname{argmin}} \int \|\nu - \mu\|^2 \mathrm{d}V \tag{5.33}$$

$$= \underset{\nu \in A}{\operatorname{argmin}} \int \left(\frac{1}{2} \|\nu\|^2 - \nu^T \mu \right) \mathrm{d}V. \tag{5.34}$$

This projection operation is a non-expansive mapping under the Euclidean metric.

Consider a vector field β whose divergence is proportional to the linear term b_1 of the pressure solve. The objective functional $F(p)$ can be readily shown to be equivalent to

$$F(p) = \frac{\Delta t^2}{\rho_{\max}} \int \left(\frac{1}{2} \|\nabla p\|^2 + b_1 p \right) \mathrm{d}V \tag{5.35}$$

$$= \frac{1}{\rho_{\max}} \int \left(\frac{1}{2} \|\mu_p\|^2 - \beta^T \mu_p \right) \mathrm{d}V, \tag{5.36}$$

where β satisfies $\frac{\Delta t}{\rho_{\max}} \nabla \cdot \beta = b_1 = (1 - \phi^{n+1}|_{p=0})$. But as

$$\phi^{n+1}|_{p=0} = \left(\phi^n + \frac{\Delta t}{\rho_{\max}} \nabla \cdot \mu_0 \right) + \frac{\Delta t}{\rho_{\max}} \nabla \cdot \mu_s, \tag{5.37}$$

this implies that β is simply a constant, say β_0, minus μ_s. Therefore, minimizing $F(p)$ is equivalent to minimizing $F'(p) = \int \|\mu_p - \beta_0 + \mu_s\|^2 \mathrm{d}V$, and the pressure solve can be expressed as a projection

$$\mu_p = P(\beta_0 - \mu_s \,;\, A_1) \tag{5.38}$$

onto the convex set $A_1 = \{\nabla p : p \geq 0\}$.

The friction solve is more straightforward. Its objective function is simply

$$E(\mathbf{s}) = \frac{1}{\rho_{\max}} \int \left(\frac{1}{2} \|\mu_s\|^2 + \mu_s^T (\mu_0 + \mu_p) \right) \mathrm{d}V, \tag{5.39}$$

so the frictional impulse is the projection

$$\mu_s = P(-\mu_0 - \mu_p \,;\, A_2) \tag{5.40}$$

onto the set $A_2 = \{\nabla \cdot \mathbf{s} : -s_{\max} \leq s_{ij} \leq s_{\max}\}$.

By substituting (5.40) into (5.38), we obtain the fixed-point property

$$\mu_p = P(\beta_0 - P(-\mu_0 - \mu_p \,;\, A_2) \,;\, A_1) \tag{5.41}$$

which characterizes the solutions to the stress response. In fact, since the right-hand side of the above is a composition of projections, it is a non-expansive mapping and is often contractive. Thus, iteratively applying the projections (5.40) and (5.38) in a staggered sequence, or equivalently, solving each quadratic program (5.28) and (5.29) in turn, is a valid method for solving the coupled system.

After the internal stresses p and $\mathbf{s}$ have been computed, it remains to integrate (5.2) and (5.3) to advance the physical state of the material to the next time step. This is described in the following section.

5.4 PARTICLE-BASED ADVECTION

Unlike traditional fluids considered in computer graphics, the flow of a granular material may not be purely incompressible due to the absence of cohesion. As a consequence, it is difficult to apply techniques based on semi-Lagrangian advection while ensuring the conservation of mass and momentum. Furthermore, a granular material that exhibits a coherent surface in a pile at rest may transition into a sparse cloud of grains in very energetic events such as splashes or free fall, rendering surface tracking methods used for animating liquids inapplicable. Therefore, a Lagrangian approach is used here for

advecting the material, which can easily handle these properties. This method can be considered as an extension of the fluid-implicit-particle (FLIP) method (Brackbill and Ruppel, 1986; Zhu and Bridson, 2005).

In the Lagrangian setting, the granular material is represented as a set of simulation particles. Each simulation particle represents not a single grain but a macroscopic sample of the material—a moving "clump" of matter with mass m_i centered at a point $\mathbf{x}_i$ and moving with average velocity $\mathbf{v}_i$. At the beginning of a time step, the continuum values of ρ^n and $\mathbf{v}^n$ are defined by accumulating the values of particles near each grid cell. Each particle is treated as a point mass, and its contribution is divided among its neighboring 8 grid nodes using standard trilinear weights. The internal stresses are then computed through the continuum model to determine the intermediate velocity $\tilde{\mathbf{v}}$. Finally, advection is performed by updating the particles using this velocity field.

In the advection step, each particle's position is updated based on the grid velocity. First the displacement $\Delta\mathbf{x}$ for density correction (section 5.3.2) is applied, then particle positions are advanced using the velocity field. Particle velocities are updated by adding the change in grid values from the previous time step, following the FLIP method.

5.4.1 PARTICLE SHAPES AND SPLIT/MERGE OPERATIONS

Unlike fluids like water which are practically incompressible, granular materials can exhibit visibly diverging flow, such as a dispersing mass of sand thrown into the air. This presents a difficulty for traditional particle-based advection techniques designed for incompressible flow: particles spread farther and farther from each other, and the simulated fluid eventually separates into clumps corresponding to individual particles instead of spreading uniformly. While this is satisfying for liquids, or for wet sand which exhibits cohesion, grains in a dispersing cloud of *dry* sand are often smoothly distributed.

To ensure that the material remains faithfully sampled in such cases, it is necessary to somehow track the spreading of simulation particles, and insert additional particles

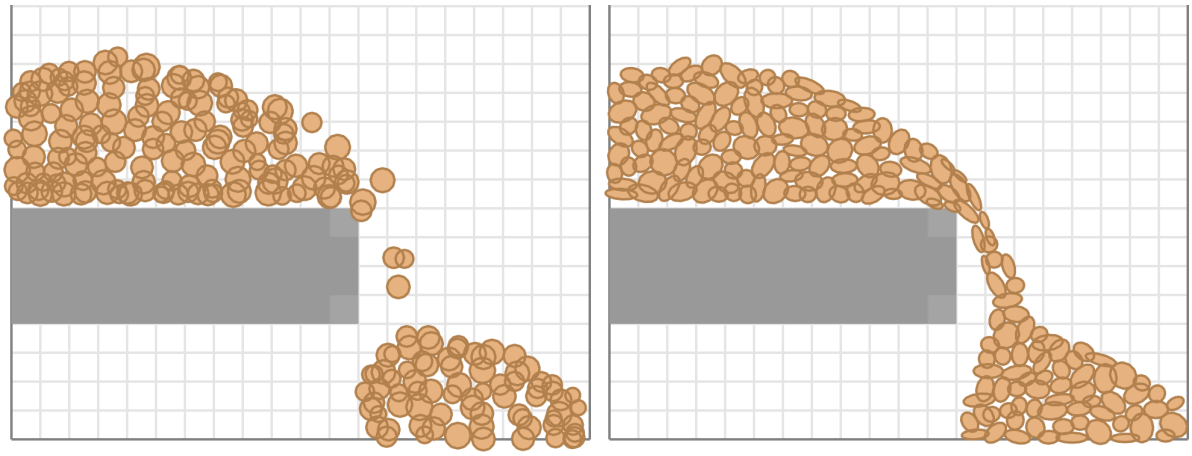


Figure 5.6: If particle distortion is not tracked (left), particles cannot remain well distributed in the material during diverging flow. By tracking the shape of particles under the flow (right), they can be split and merged appropriately to maintain a good distribution.

where necessary. This can be done simply by attaching to each particle a shape centered around the particle position, initially spherical, which is stretched and squeezed by the flow. When the particle becomes too large or too small, it is split into two or merged with an adjacent particle, thus automatically maintaining a good distribution of particles (figure 5.6). Previous work in incompressible SPH-based fluid simulation has used splitting and merging as a level of detail approach to accelerate simulation (Adams et al., 2007), but here it is a necessity for representing a dispersing mass of not-incompressible material.

In general, the shape of a particle is an ellipsoid, which can be represented as the region $(\mathbf{x} - \mathbf{x}_i)^T \mathbf{A}_i^{-1} (\mathbf{x} - \mathbf{x}_i) \leq 1$ for a symmetric positive definite matrix $\mathbf{A}_i$. The semi-axes of the ellipsoid are given by the eigenvectors and square roots of eigenvalues of $\mathbf{A}_i$. To first order, as this ellipsoid is advected through the velocity field $\mathbf{v}$, its time evolution is given by

$$\frac{\mathrm{d}\mathbf{A}_i}{\mathrm{d}t} = \mathbf{A}_i \mathbf{J}_{\mathbf{v}}^T(\mathbf{x}_i) + \mathbf{J}_{\mathbf{v}}(\mathbf{x}_i)\mathbf{A}_i \tag{5.42}$$

where $\mathbf{J}_{\mathbf{v}}(\mathbf{x}_i)$ is the Jacobian of the velocity field at position $\mathbf{x}_i$.

A user-specified parameter r controls the size of particles. A particle is defined as *valid* if the lengths of its semi-axes lie within the range $[\frac{1}{\sqrt{2}}r, \sqrt{2}r]$. Experiments suggest

that setting r to one-fourth the grid spacing is generally effective. The bounds on the axis lengths prevent particles from becoming too big, too small, or too skinny, so that they can be treated as points when interpolating grid values. Split and merge operations are applied to particles that become invalid. These operations are defined so that they conserve the mass, momentum, and center of mass of the system.

A split operation divides a particle into two identical particles along the longest axis. Each child particle has half the axis length along this direction, while the other two axes remain the same.

A merge operation replaces two nearby particles with one larger particle centered at their center of mass, carrying their total mass and momentum. Upon merging particles i and j, the shape of the new particle, say k, is given by

$$m_k \mathbf{A}_k = m_i(\mathbf{A}_i + \Delta\mathbf{x}_i \Delta\mathbf{x}_i^T) + m_j(\mathbf{A}_j + \Delta\mathbf{x}_j \Delta\mathbf{x}_j^T), \tag{5.43}$$

where $\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_k$, and similarly $\Delta\mathbf{x}_j$, are the displacements of the old particles from the new center of mass. This rule ensures that immediately merging the children of a split results in the original particle.

Split and merge operations are performed only when the resulting particle(s) are valid. In practice, this yields a consistent set of particles without oscillations.

5.5 IMPLEMENTATION DETAILS

An overview of all the steps of the method is shown in figure 5.7. Below I describe some implementation details.

For the continuum model, a regular Cartesian grid is used, on which physical quantities are stored in a "staggered" fashion following Goktekin et al. (2004). Scalars ρ and p and diagonal components of $\mathbf{s}$ are stored at cell centers, vector components of $\mathbf{v}$ at cell faces, and off-diagonal components of $\mathbf{s}$ at cell edges (see figure 5.8). Spatial derivatives are computed through first-order finite differences. For stability, time steps are chosen so that

For each time step:

1. Accumulate density ρ^n and velocity $\mathbf{v}^n$ onto grid.
2. Perform density correction on ρ^n.
3. Repeat until convergence or maximum iterations:
   a) Compute friction $\mathbf{s}$ by minimizing (5.20) for fixed p.
   b) Compute pressure p by minimizing (5.13) for fixed $\mathbf{s}$.
4. Find intermediate velocity $\tilde{\mathbf{v}}$ through (5.5).
5. Update solid bodies with an impulse $\Delta t\mathbf{F}$ (5.23).
6. Update particles:
   a) Update velocities using FLIP.
   b) Move particles through the velocity field $\tilde{\mathbf{v}}$.
   c) Update shapes using (5.42).
   d) Split and merge particles.

Figure 5.7: The main steps of the method.

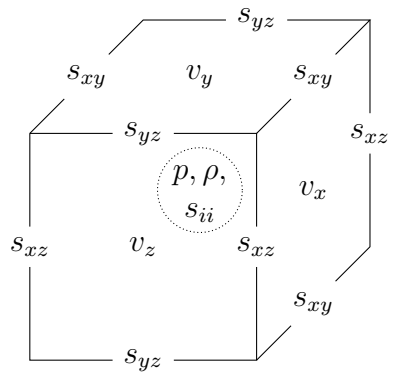


Figure 5.8: The staggered grid for storing scalars p and ρ, vector field $\mathbf{v}$, and tensor field $\mathbf{s}$.

no particle moves more than half of a grid cell in a single time step. As the impulse-based integration scheme means that forces only act on particles at the beginning of a time step, a forward Euler step suffices to perform particle advection.

In the discretized setting, (5.13) and (5.20) are quadratic programs (QPs) with sparse, symmetric, positive semidefinite matrices. As in the previous chapter, these problems can

be solved efficiently using the recent algorithm of Dostál and Schöberl (2005), which can be extended to support two-sided bound constraints. However, a naïve approach causes the friction solve to converge slowly, because of the coupling between different components of $\mathbf{s}$. By instead minimizing with respect to one component of $\mathbf{s}$ at a time (which amounts to optimizing over orthogonal subspaces in turn), an approximate frictional solution can be found which converges much more rapidly. Because the projections are nevertheless repeated in an outer loop, the correct solution remains the fixed point of the procedure. More details are given in the following subsection.

Convergence is also greatly accelerated by warm-starting the solver using the pressure and friction values computed at the previous time step as initialization. In practice, performing just a few iterations of staggered projections sufficed to give stable and convincing results; only 2 iterations were used in all the results shown.

5.5.1 SEPARATION OF COMPONENTS

Solving the friction projection (5.40) involved a matrix D_2 representing the tensor gradient $\nabla \cdot \mathbf{s}$, which for a trace-free tensor field $\mathbf{s}$ has 5 times the dimension of the corresponding to the gradient of a scalar p. Directly solving the quadratic program using this full system leads to numerical difficulties and poor convergence. Instead, it is more efficient to perform minimization on each component of $\mathbf{s}$ in turn.

That is, first one minimizes E with respect to the component s_{yy}, say, holding all other components fixed. The minimization is then of the form

$$E = E|_{s_{yy}=0} + \frac{1}{2}\mathsf{s}_{yy}^T \mathsf{A}_{yy} \mathsf{s}_{yy} + \mathsf{b}_{yy}^T \mathsf{s}_{yy}, \tag{5.44}$$

$$\mathsf{A}_{yy} = \frac{\Delta t^2}{\rho_{\max}} \mathsf{D}_y^T \mathsf{D}_y, \tag{5.45}$$

$$\mathsf{b}_{yy} = \frac{\Delta t}{\rho_{\max}} \mathsf{D}_y^T \rho^n \tilde{\mathbf{v}}|_{s_{yy}=0}. \tag{5.46}$$

This simply involves the finite difference matrix D_y corresponding to ∂/∂_y, which is numerically much more well-behaved. (Note that in the absence of solid bodies, the

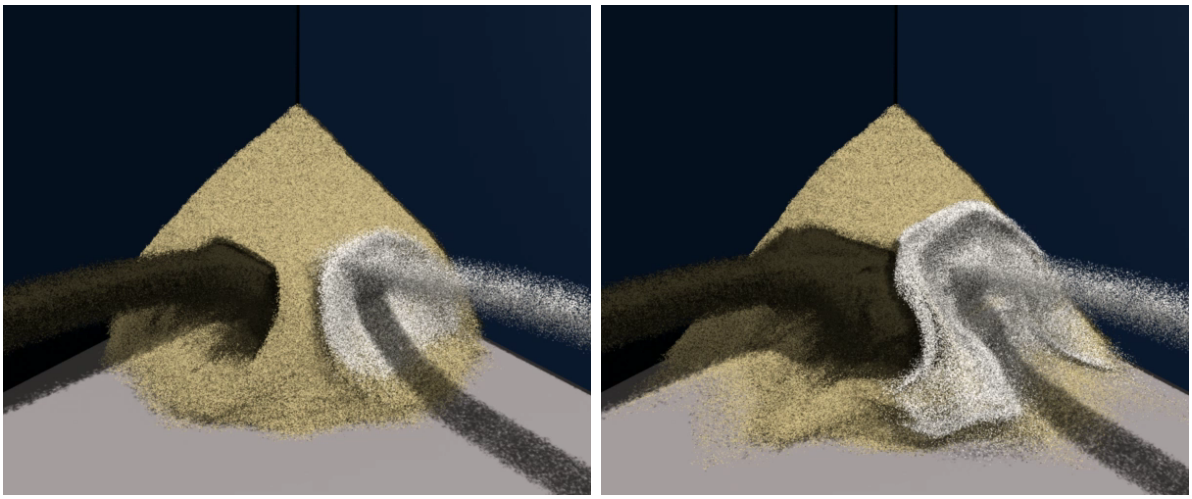


Figure 5.9: An example of multiple granular materials interacting in a scene. From left to right: high density and friction (black), medium density and friction (sandy), low density and friction (white).

components of s_{yy} at different x and z positions are decoupled, and each column can be solved independently. This does not hold when interacting solid bodies are present.) After this, s_{xx} and s_{zz} are solved together, using the trace-free condition $s_{xx} + s_{zz} = -s_{yy}$. Similarly the diagonal components s_{xy}, s_{yz} and s_{xz} are determined in turn, holding previously solved components fixed at their updated values.

This amounts to performing minimization over a set of orthogonal subspaces that span the space of frictional stresses. Since each minimization is a projection, and the coupled solution is a fixed point of each of them, a staggered projection sequence that solves for p, s_{yy}, s_{xx} & s_{zz}, s_{xy}, s_{yz}, and s_{xz} in turn remains a non-expansive mapping with the coupled solution to (5.28) and (5.29) as its fixed point.

5.5.2 MULTIPLE INTERACTING MATERIALS

This method can easily be extended to handle multiple granular materials with different properties interacting in a single scene (figure 5.9).

To account for materials of varying densities and friction coefficients, each simulation particle i is associated with its own values of $\rho_{\max,i}$ and α_i. For the pressure constraint,

the volume fraction ϕ must be redefined because $\rho_{\max}$ will vary for materials of different densities. Instead of accumulating m_i of each particle on the grid to obtain ρ and computing $\phi = \rho/\rho_{\max}$, one can obtain ϕ directly by accumulating the minimum volume of each particle $V_{\min,i} = m_i/\rho_{\max,i}$. The pressure solve is then given by

$$\mathsf{A}_1 = \Delta t^2 \mathsf{D}_1^T \frac{\phi^n}{\rho^n} \mathsf{D}_1, \tag{5.47}$$

$$\mathsf{b}_1 = 1 - \phi^{n+1}|_{p=0}, \tag{5.48}$$

instead of (5.10) and (5.11). Here ϕ^n/ρ^n is treated as a diagonal matrix.

In the friction solve, the weighting w now simply equals ϕ. This leads similarly to replacing (5.18) and (5.19) with

$$\mathsf{A}_2 = \Delta t^2 \mathsf{D}_2^T \frac{\phi^n}{\rho^n} \mathsf{D}_2, \tag{5.49}$$

$$\mathsf{b}_2 = \Delta t \mathsf{D}_2^T \phi^n \tilde{\mathbf{v}}|_{\mathbf{s}=0}. \tag{5.50}$$

Since the coupling matrices of section 5.3.4 are computed independently of the material, they remain exactly the same. In the particle-based advection part, two particles are only merged if they have identical material properties. This ensures that each simulation particle is associated with only one distinct material, and facilitates rendering. If it is desired to simulate materials with properties that vary continuously over space, this condition can be relaxed.

5.6 RESULTS AND DISCUSSION

This method has been applied to several scenarios, showing many characteristic behaviors of granular materials including stable pile formation, freely dispersing clouds of grains, and two-way rigid body interaction.

Absence of cohesion: In figure 5.1, several rigid bodies are placed on top of a sand pile, and an explosion goes off inside the pile, sending the sand and the bodies into the

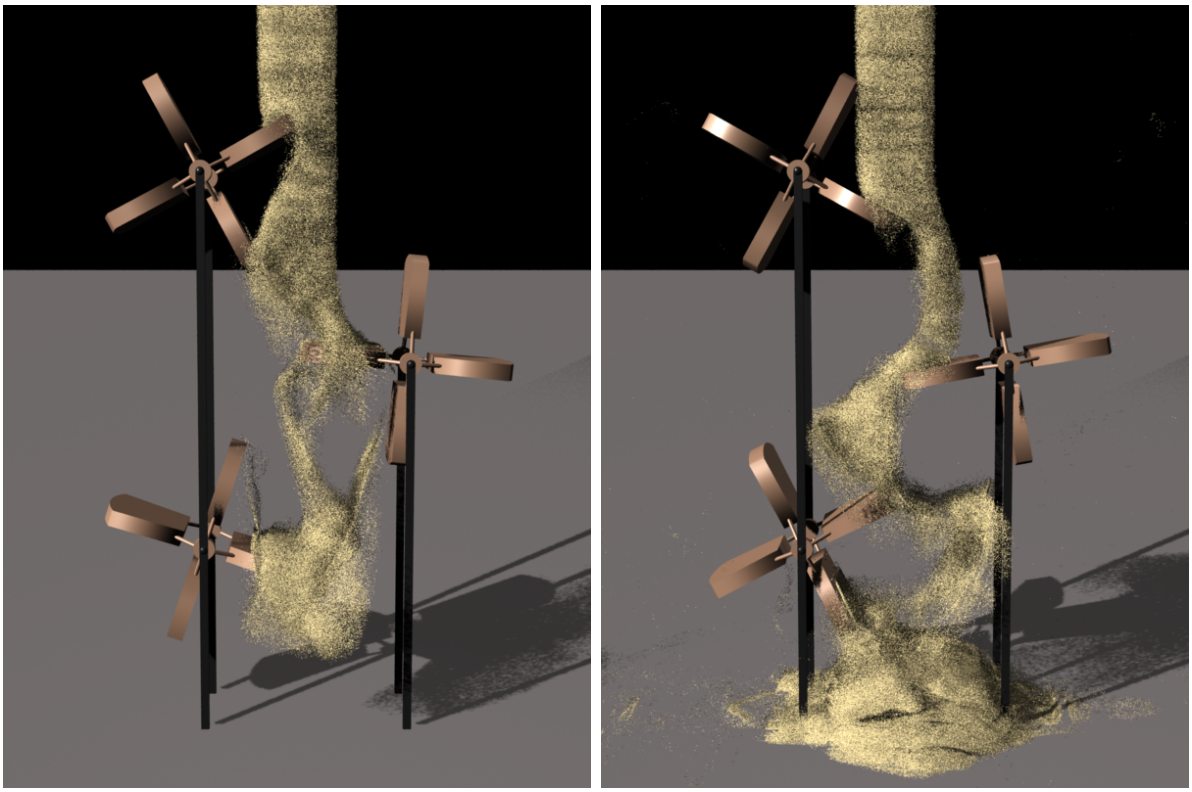


Figure 5.10: Sand falls on a series of paddle wheels, setting them in motion.

air. The explosion was modeled as an instantaneous outward impulse applied to particles in a small sphere in the interior of the pile. In an incompressible fluid, this divergence would be immediately nullified by the pressure projection, but the cohesionless pressure solve allows the sand to disperse in a realistic manner.

Pressure/friction interaction: Figure 5.3 shows sand falling in an hourglass. Friction plays a central role here, as it maintains a constant rate of flow of sand through the neck of the hourglass, unlike a traditional liquid whose rate of flow would depend on the height of the liquid above it.

Solid coupling: Figure 5.5 demonstrates two-way coupling between rigid bodies and granular material. Note that an accurate coupling of frictional stress is necessary for the material to be able to support the weight of bodies much denser than itself. Friction also

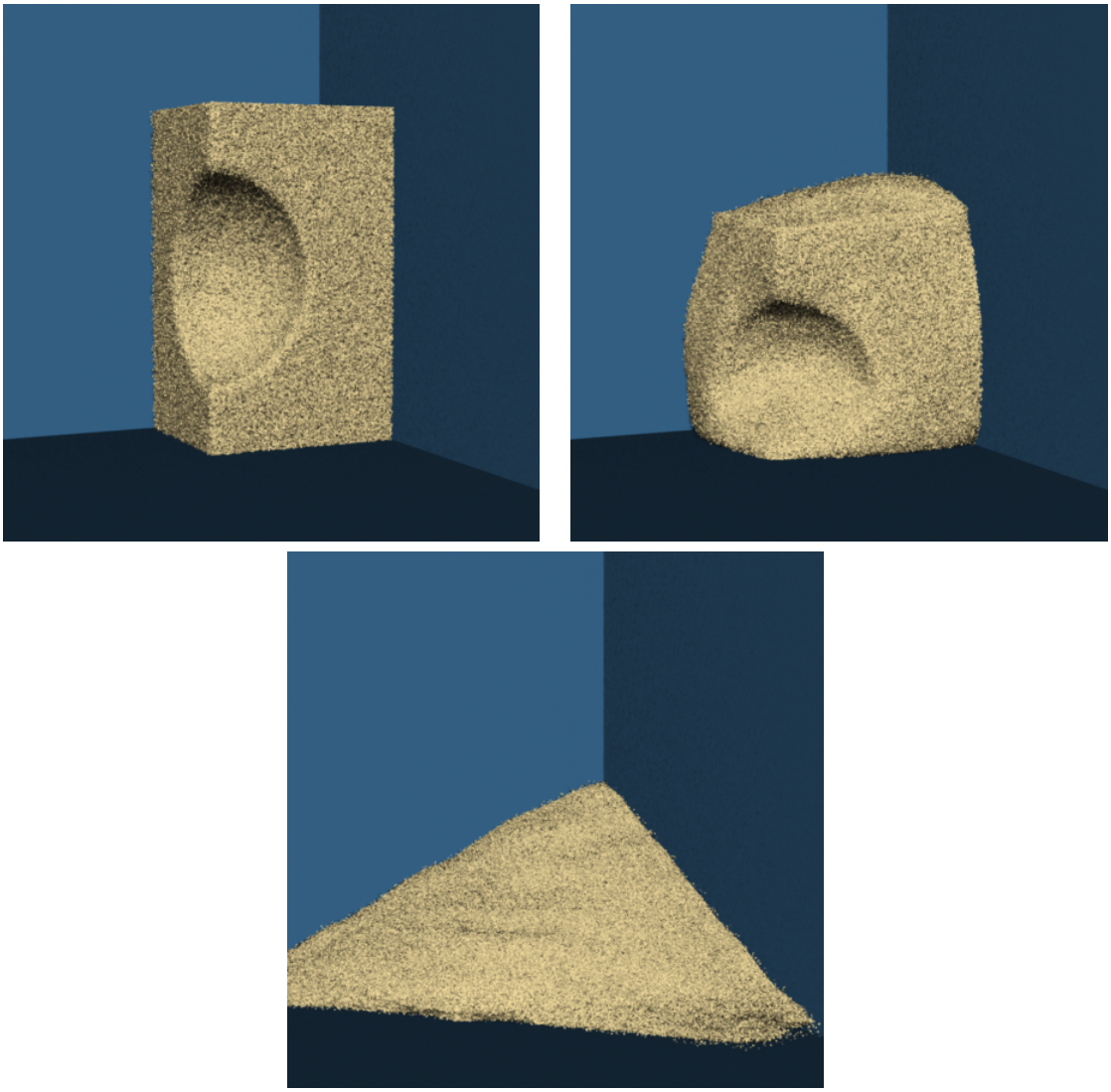


Figure 5.11: A column of granular material is simulated using the presented method.

causes the lighter spheres roll without slipping on the sand surface. Another example with rotating paddle wheels is shown in figure 5.10.

Multiple materials: Figure 5.9 shows different kinds of granular materials colliding and interacting in a single scene. The differing densities and friction of the materials give rise to different interactions.

Comparison with previous work: In figure 5.11, the behavior of the proposed technique is shown on initial conditions identical to figure 6 of Zhu and Bridson (2005).

Due to their incompressibility assumption, their approach works well for modeling cohesive materials like wet sand, while the simulator presented here behaves more like a dry granular material with zero cohesion.

Real-world comparison: Figure 5.12 shows an impact scenario with a fast-moving metal sphere. This was modeled on a real experiment from the Discovery Channel, viewable at http://dsc.discovery.com/videos/time-warp-deep-impact.html. The result is qualitatively consistent with the real-world behavior of sand in this scenario. Some anisotropy visible in the splash is due to the linearization of the friction constraint (5.15). This can be avoided by adding more constraint hyperplanes, at the cost of higher simulation time.

Rendering granular materials from a continuum-based simulation poses its own challenges, as fine grains for rendering must be sampled from the simulation in a temporally coherent manner. Previous methods either simulated a cohesive material and maintained a well-defined surface for rendering (Zhu and Bridson, 2005), or attached grains rigidly to simulation particles (Lenaerts and Dutré, 2009) leading to visible clumps. Neither of these is applicable to the current technique, necessitating a heuristic approach that was found to perform well for many situations.

In the current implementation, sand is rendered as a cloud of points, producing a granular appearance. To each simulation particle are associated a number of render points sampled within the particle's ellipsoid, that are passively advected with the flow. Points are reassigned to new simulation particles upon split and merge events, and resampled if they fall outside their parent ellipsoid. The number of points being rendered can be reduced by detecting connected regions of high density and avoiding sampling points inside them. For illumination, the point normal is taken to be the gradient of the density plus a per-particle random jitter. All scenes were rendered using Pixar's RenderMan®.

The performance of a single-threaded implementation was measured on a 3.33 GHz

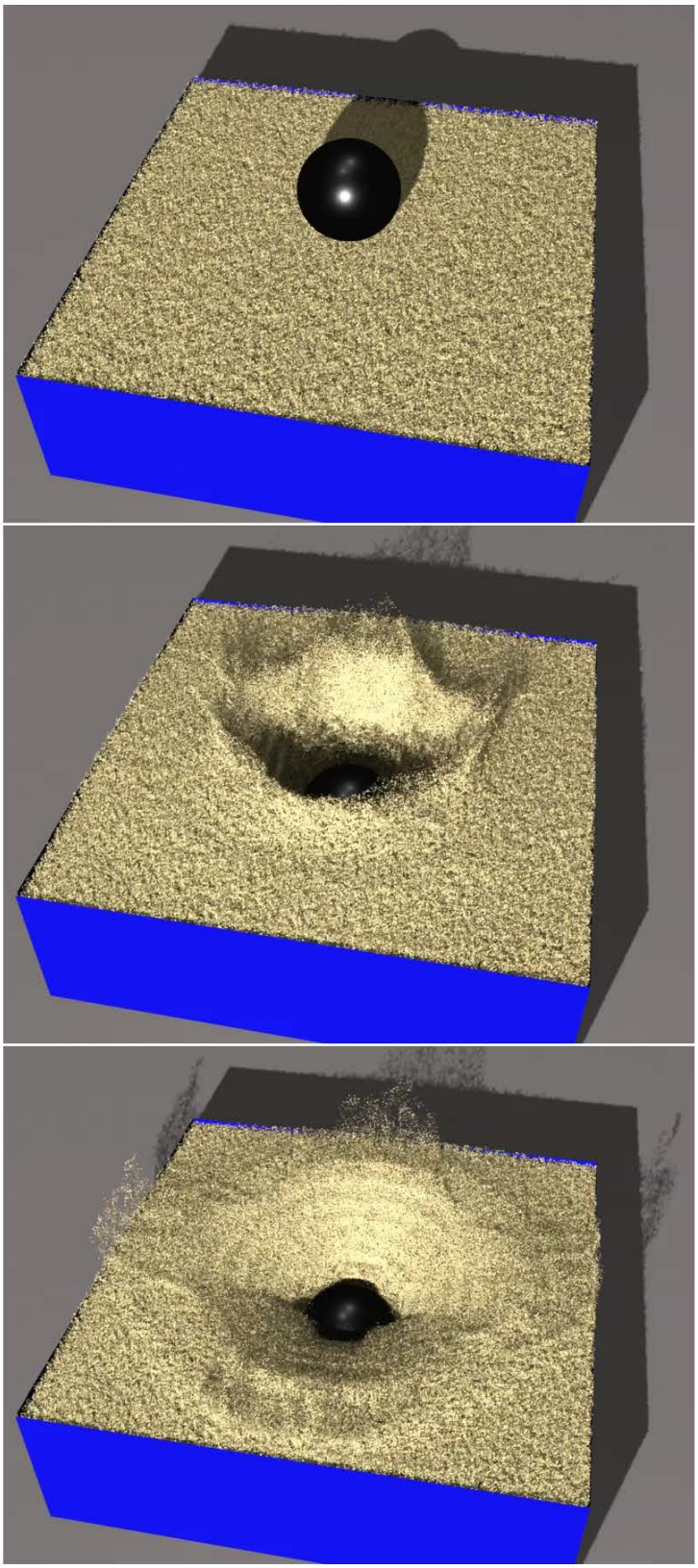


Figure 5.12: A 1-inch metal sphere hits sand at high velocity, creating a splash and a large crater.

Example	Grid size	Sim/render particles	Time/ frame
Explosion (Fig. 5.1)	$75 \times 50 \times 75$	242k / 4.7M	19.5 s
Hourglass (Fig. 5.3)	$25 \times 50 \times 25$	44k / 2.4M	6.5 s
Impacts (Fig. 5.5)	$50 \times 40 \times 100$	425k / 4.1M	18.3 s
Paddles (Fig. 5.10)	$50 \times 100 \times 50$	199k / 2.7M	32.5 s
Materials (Fig. 5.9)	$60 \times 40 \times 60$	415k / 4.4M	13.9 s
Comparison (Fig. 5.11)	$75 \times 50 \times 75$	506k / 4.0M	11.6 s
Splash (Fig. 5.12)	$50 \times 40 \times 50$	403k / 5.2M	6.1 s

Table 5.1: Performance measurements for all of the examples shown.

Intel Core i7 machine with 5.8 GB of RAM. The detailed performance numbers are shown in Table 5.1. On average, the time per frame was spent as follows: 17% pressure, 35% friction, 33% particle update, and the remainder in other steps.

All of the simulations took between 6 and 33 seconds per frame on average, which is comparable to the performance of Zhu and Bridson (2005) on similar PCs. The simulation time per frame is proportional to both the number of occupied grid cells and the speed of motion in the scene due to adaptive timestepping. In comparison to Bell et al. (2005), their hourglass scenario with 110k particles and rigid body impact ("splash") with 187k particles both take roughly 200 seconds per frame. Accounting for the scaling factor in the number of particles, this method running on the similar scenes and hardware is about one order of magnitude faster.

5.6.1 LIMITATIONS AND FUTURE WORK

The method described here models purely cohesionless behavior, which is an assumption satisfied by most dry granular materials. However, certain materials such as wet sand and soil show a finite amount of cohesion, which prevents the material from dispersing under small forces. Existing continuum models (Zhu and Bridson, 2005; Lenaerts and Dutré, 2009) which assume incompressibility can model such materials to an extent, but do not handle cases when cohesive forces are overcome and the material breaks apart. An approach that can faithfully model granular materials with varying amounts of cohesion

remains an open challenge.

Currently, the detailed effects of inter-grain interaction are not modeled. These include subgrid-scale variations in motion, and interactions between grains of widely differing sizes. The addition of a model for such subgrid-scale interactions would allow for the simulation of even more complex scenes such as avalanches involving objects of many different shapes and sizes. Such an approach could also be useful in engineering applications.

To derive a tractable model for granular material dynamics, a critical state assumption is adopted, and inter-particle interactions in the collisional regime of lower density are neglected. These assumptions allow many scenarios of interest to graphics applications to be simulated efficiently, but preclude modeling the more counter-intuitive behaviors such as formation of convection layers and surface waves upon shaking, and the Brazil nut and reverse Brazil nut effects. These surprising phenomena are beyond the scope of the current work, and remain as puzzling, challenging effects to model for the physics and mathematics communities.

Finally, the issue of sampling and rendering millions of grains from a continuum representation of granular material is an interesting research problem in itself, and can further enhance the visual appearance of this technique. Further independent investigation of this problem is valuable.

CHAPTER 6

CONCLUSION

As the variety of examples presented in this dissertation indicates, multiscale phenomena are ubiquitous in the real world. In fact, most complex systems composed of a hierarchical structure are multiscale in nature, and in such phenomena one can see fine, dynamic detail embedded in and interacting with a large-scale structure. For graphics applications, simulating such phenomena is of growing importance, since improving the realism of virtual scenes will depend on capturing the visual richness of such complex, detailed phenomena. At the same time, such simulation problems have remained very challenging, as traditional techniques become prohibitively expensive when faced with wide ranges of scales.

In this dissertation, I have proposed that it is possible to visually model and simulate such multiscale phenomena practically and efficiently on current commodity hardware. This is enabled by decomposing the phenomena into macroscopic and microscopic scales of global and local motion; choosing separate, efficient models for the dynamics at these different scales; and coupling these models together to reflect the interactions between scales. This is a very general approach that can be applied to many different kinds of multiscale phenomena. Its computational benefit comes from enabling one to choose, at each scale of the phenomenon, the mathematical model that most efficiently captures the behavior of interest at that scale.

As this approach is a broad conceptual framework rather than a specific computational

recipe, its application to particular problems can require new developments. In some cases, such as with the work on fluids presented in this dissertation, there are excellent existing models for both the macroscopic and the microscopic features, but each of them work in isolation. Here, novel techniques are needed to couple these two models together in a way that is consistent with the behavior of the multiscale system. In other cases, only expensive microscale models are known in previous work, such as per-agent crowd simulation or discrete methods for granular materials. Then it is necessary to develop entirely new coarsened representations to which the expensive computations can be mapped, thereby simplifying the microscopic representation and accelerating the simulation.

The four applications described in this dissertation demonstrate the benefits of such an approach. Below, I summarize the main results specifically related to these applications.

6.1 SUMMARY OF RESULTS

I presented two techniques for animating the flow of highly detailed fluids, coupling numerical simulation with texture synthesis and procedural modeling of turbulence. I also presented two techniques for modeling pedestrian crowds and granular materials, using a traditional discrete representation for the fine-scale objects and introducing a continuum representation for the overall aggregate flow. All these techniques provide a significant performance benefit of about an order of magnitude when compared to previous techniques that show a similar level of detail.

The technique of texture synthesis for detailed fluid animation involves two major contributions. First, I introduced a novel scheme for coupling the synthesis of heterogeneous texture to the characteristics of the large-scale flow, enabling the variation of surface detail to be correlated directly to the physical behavior of the fluid. Second, I described a way to model the intrinsic dynamics of the fluid detail by capturing the temporal variation of texture from footage of real fluids, producing a dynamic, evolving

surface appearance on the simulated fluid. With this technique, a variety of fluids with complex surface appearance such as breaking waves, river rapids, and flowing lava can be modeled easily to produce visually compelling results.

The work on procedural modeling of turbulence introduces several new ideas to the field of computer graphics. This was one of the first techniques to model both the fine-scale turbulent behavior and the large-scale flow of fluids in general scenarios including interaction with obstacles. Here I proposed a model to describe the transfer of energy from the mean flow to the turbulent vortices and track the distribution of turbulent energy over space and over scales. I also introduced an advected noise scheme to generate temporally coherent procedural noise that moves with the fluid, which is a necessity for generating smooth turbulent flow. Finally, I described ways to couple the generated turbulence to the numerically computed mean flow and to extend the method to free-surface flows. This technique yields realistic animation of highly detailed turbulent flow at a performance of 5 to 20 times faster than purely numerical simulation at a similar level of detail.

For simulating the motion of humans in dense crowds, the technique I presented augments a traditional discrete model of individual agents with a continuum representation of the aggregate flow of the crowd. In the continuous domain, I introduced a novel variational constraint called unilateral incompressibility, which is a general formulation to describe the macroscopic flow of large collections of objects. This efficiently models inter-agent collision avoidance in densely crowded scenarios. At the agent level, knowledge of the continuum flow allows local planning for individuals to be greatly simplified, leading to an extrememly fast model for the motion of individual agents. This approach allows large crowds containing hundreds of thousands of agents to be simulated on commodity desktop PCs at near-interactive rates. In particular, the technique is highly scalable and runs ten times faster than contemporary per-agent crowd simulation models.

Finally, I presented a novel and highly efficient technique for simulating granular materials such as sand, that builds upon the crowd simulation model and adds several

new contributions. Here, the unilateral incompressibility model was applied to compute the internal pressure of the material, allowing free flow without artifical cohesion. I also presented a numerical scheme for robust and efficient computation of the frictional stresses. All the internal stresses are computed together in a coupled fashion, which enables a wide range of granular flow behaviors to be realistically captured. In particular, this is the first continuum model which supports two-way interaction with rigid bodies. To couple this continuum representation with a model of discrete grains, I introduced techniques for the advection, sampling, and rendering of such a granular system. This approach is as fast as previous continuum-based methods, but reproduces much more of the unique physical behavior of granular materials. Relative to previous discrete methods which show comparable fidelity, the computational cost of this technique is an order of magnitude lower, bringing the computational cost down from minutes to seconds per frame.

6.2 LIMITATIONS

For each of the techniques presented in this dissertation, I have discussed several limitations in the corresponding chapters. Here, I summarize the key limitations of each. All of these techniques based on scale decomposition rely on exploiting simplified computational models for fine-scale details. As such, the fidelity of the results depends on how well the simplification captures the true microscale behavior. Exceptional scenarios which violate the corresponding simplifying assumptions can lead to unrealistic results. This is a source of error common to all these methods.

One limitation of the texture synthesis approach to fluid detail is that it requires the user to manually specify the correspondence between the fluid behavior and the variation of surface texture. This involves both choosing a parametrization of the heterogeneous texture input and defining the relationship between fluid features and the texture parameter map on the fluid surface. While this provides a high degree of

artistic control, a more automatic approach may be desirable for certain applications. Another major limitation is that representing detail through texture limits the approach to only modifying the visual appearance of the surface. It is difficult to use texture to add true three-dimensional detail to the motion of the fluid or to the surface geometry in completely general scenes.

The method for procedural modeling of turbulence was developed to overcome both of these limitations. However, in the present work, it relies on a direct application of the Kolmogorov model which assumes that the turbulent component of the flow is isotropic, locally homogeneous, and spatially uncorrelated. While these assumptions hold well for fully developed turbulence, they are violated in some scenarios such as at the transition from smooth laminar flow to turbulence, at boundary layers near obstacles, and at free surfaces. In such cases, the turbulent vortices themselves have certain long-range correlations which are not captured in the proposed energy model and can lead to visible artifacts in the results. Inspired by the work reported here, there has been some subsequent by other authors to address these extend the turbulence model to boundary layers (Pfaff et al., 2009), free-surface turbulence (Kim et al., 2009), and anisotropy (Pfaff et al., 2010), which address some of these limitations.

Questions also remain about how much the scales of procedural turbulence and the simulated flow should overlap. This depends closely on the amount of artificial dissipation in the numerical simulation that must be counteracted by the turbulence model; for very non-dissipative numerical schemes as proposed in recent work (Kim et al., 2007b; Selle et al., 2008; Kim et al., 2008a; Mullen et al., 2009), it may be sufficient for the turbulent vortices to remain at the purely subgrid scale. Further, it is an open question whether the combined spectrum of the large-scale and turbulent flow follows the appropriate distribution. These experiments are left for further investigation.

In crowd simulation, the continuum model is designed specifically for local collision avoidance. This choice leads to certain limitations of the current work, when other inter-

agent interactions become significant. One example is that the unilateral incompressibility criterion depends only on local information, and so cannot anticipate future collisions from distant agents. Thus, two groups of agents approaching each other will not react until they are adjacent to each other. This is a case where interactions between microscopic entities take place over large macroscopic distances. As another example, human crowds in real life also display behavioral features other than purely navigating towards the goal. For example, there is often an informal cultural convention to walk on one side of the path to avoid oncoming pedestrian traffic. Such cultural and social behaviors would affect the dynamics of the crowd, and would entail changes in both the per-agent and continuum models presented here.

For granular materials, the presented technique adopts a critical state assumption, and does not model the detailed effects of inter-grain interaction. Effectively, it is assumed that all grains move coherently with the large-scale flow. Thus, subgrid-scale variations in motion and interactions between grains of widely differing sizes are neglected in the current work. Such interactions give rise to highly counter-intuitive behaviors of granular materials, particularly in highly energetic scenarios when the material is agitated heavily; these include the formation of convection layers and surface waves, and the Brazil nut and reverse Brazil nut effects. These phenomena are outside the scope of the current work and remain under active investigation in the physics and mathematics communities (Behringer et al., 1999).

6.3 FUTURE WORK

The approach of scale decomposition presented in this dissertation, coupling numerical methods for coarse scales with simplified models for fine detail, is a novel class of techniques whose applications to visual simulation have only begun to be explored. There are many exciting areas for further work to be done using this approach, both in developing its further applications, and in formalizing its theoretical basis. I have described in each

chapter some of the avenues of future work with regards to each specific application. Here, I discuss some of the broader possibilities and open problems.

Many of the phenomena we see all around us in nature and in society are multiscale, being composed of much finer constituent parts that interact to give rise to large-scale behavior. In this dissertation, four specific examples of such phenomena were addressed to demonstrate the versatility of an approach that directly engages the multiscale nature of these problems. There is still an enormous space of other multiscale phenomena that would benefit from such techniques. I describe a few of these of particular interest to graphics and animation below.

Biological structures are a common example. The behavior of skin and muscles arises from the interaction of multiple layers of tissues with very different physical properties and shows both gross deformations and fine wrinkles in a highly correlated fashion. The motion of hair is another phenomenon where a large number of hair strands in fine-scale frictional and cohesional contact give rise to the appearance of a continuous flowing mass. Closely related is the problem of cloth simulation, where it is desirable to capture a wide range of features from detailed wrinkles at small scales to the volumetric interactions between multiple layers of clothing. A scale decomposition approach for these problems would enable efficient, interactive simulation of highly realistic characters.

The motion of large aggregate materials composed of objects of varying properties, such as size, shape, stiffness, and strength, is a computational challenge for a number of applications. These include the modeling of such natural disasters as landslides and avalanches, and simulation of large piles of objects in computer graphics. Present methods for simulation of heterogeneous collections of materials are limited to expensive computation of pairwise interaction forces between objects (the discrete element method; see Bićanić (2004)), as there is no comprehensive theoretical framework of their aggregate dynamics. There is an enormous potential for computational benefits by developing new techniques that would abstract the inter-object forces into a continuum representation.

The proposed framework offers a great opportunity for combining purely synthetic models, such as numerical simulation and procedural modeling, with information from the real world, such as scientific observations and data-driven models. The popularity of data-driven techniques has been growing steadily in computer graphics, as they allow for realistically capturing phenomena that are *a priori* difficult to model synthetically, such as realistic materials and character motion. However, they have traditionally been difficult to adapt to highly dynamic scenarios, where the system can easily depart from the space of sampled inputs. For such simulation problems, coupling numerical and data-driven methods would provide a versatile and highly accurate technique. While texture synthesis for fluids is an example of such a data-driven model for the visual *appearance* of a multiscale phenomenon, integration of data-driven models for *dynamics* remains a problem to be addressed.

To fully realize the potential of these future applications, a general theoretical framework for the design and evaluation of such techniques is needed. At present, a solution technique for a particular multiscale problem must be developed in an application-dependent fashion, by combination of certain large-scale and small-scale models through some form of coupling. The approach lacks general principles to determine how best to couple two given models together, and it is not clear how to analyze the error of the resulting model. Further research is needed to investigate the fundamental principles underlying the success of this approach and place it on a firm theoretical footing. This will provide a sound basis for the development of accurate and computationally efficient modeling techniques for many more novel simulation problems.

REFERENCES

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph*, 26(3):48, 2007. URL http://doi.acm.org/10.1145/1276377.1276437.

Iván Alduán, Ángel Tena, and Miguel A. Otaduy. Simulation of high-resolution granular media. In *Proc. of Congreso Español de Informática Gráfica*, 2009. URL http://www.gmrv.es/Publications/2009/ATO09.

Steven S. An, Theodore Kim, and Doug L. James. Optimizing cubature for efficient integration of subspace deformations. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 165:1–165:10, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1457515.1409118. URL http://doi.acm.org/10.1145/1457515.1409118.

Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In Demetri Terzopoulos and Victor Zordan, editors, *ACM SIGGRAPH /Eurographics Symposium on Computer Animation*, pages 87–96, Los Angeles, California, 2005. Eurographics Association. ISBN 1-59593-198-8. doi: 10.2312/SCA/SCA05/087-096. URL http://www.eg.org/EG/DL/WS/SCA/SCA05/087-096.pdf.

Igor S. Aranson and Lev S. Tsimring. Continuum description of avalanches in granular media. *Phys. Rev. E*, 64(2):020301, Jul 2001. doi: 10.1103/PhysRevE.64.020301.

Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM*

Trans. Graph., 21:483–490, July 2002. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/566654.566606. URL http://doi.acm.org/10.1145/566654.566606.

Michael Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, pages 217–226, 2001. URL citeseer.ist.psu.edu/530201.html.

Jernej Barbič and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 982–990, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1186822.1073300. URL http://doi.acm.org/10.1145/1186822.1073300.

Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, and John A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25:19–38, January 2006a. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1122501.1122503. URL http://doi.acm.org/10.1145/1122501.1122503.

Adam W. Bargteil, Funshing Sin, Jonathan E. Michaels, Tolga G. Goktekin, and James F. O'Brien. A texture synthesis method for liquid animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Sept 2006b.

Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100, 2007. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1276377.1276502.

O. B. Bayazit, J.-M Lien, and N. M. Amato. Better group behaviors in complex environments with global roadmaps. *Proceedings of the 8th International Conference on Artificial Life*, pages 362–370, 2002.

Robert Behringer, Heinrich Jaeger, and Sidney Nagel. Introduction to the focus issue on granular materials. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(3):509–510, 1999. doi: 10.1063/1.166426. URL http://link.aip.org/link/?CHA/9/509/1.

Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based simulation of granular materials. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer animation*, pages 77–86. ACM, 2005. ISBN 1-7695-2270-X. doi: http://doi.acm.org/10.1145/1073368.1073379.

Kiran S. Bhat, Steven M. Seitz, Jessica K. Hodgins, and Pradeep K. Khosla. Flow-based video synthesis and editing. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23 (3), August 2004.

Nenad Bićanić. Discrete element methods. In Erwin Stein, René de Borst, and Thomas J.R. Hughes, editors, *Encyclopedia of Computational Mechanics*, volume 1, chapter 11. Wiley, 2004.

Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1): 75–104, 1996.

J U Brackbill and H M Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.*, 65(2):314–343, 1986. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(86)90211-1.

Robert Bridson. Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 22, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1278780.1278807.

Robert Bridson and Matthias Müller-Fischer. Fluid simulation: SIGGRAPH 2007 course notes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 1–81, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1281500.1281681.

Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Trans. Graph*, 26(3):46, 2007. URL `http://doi.acm.org/10.1145/1276377.1276435`.

B. Chanclou, A. Luciani, and A. Habibi. Physical models of loose soils dynamically marked by a moving object. In *Computer Animation '96 Proceedings*, pages 27–35, Jun 1996. doi: 10.1109/CA.1996.540485.

S. Chenney. Flow tiles. *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 233–242, 2004.

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan Richard Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In Michael Gleicher and Daniel Thalmann, editors, *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, San Diego, California, USA, August 2-4, 2007*, pages 219–228. Eurographics Association, 2007. URL `http://doi.acm.org/10.1145/1272690.1272720`.

Robert L. Cook and Tony DeRose. Wavelet noise. *ACM Trans. Graph.*, 24(3):803–811, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073264.

O. C. Cordeiro, A. Braun, C. B. Silveria, S. R. Musse, and G. G. Cavalheiro. Concurrency on social forces simulation model. *First Intl. Workshop on Crowd Simulation*, 2005.

Frederic Cordier and Nadia Magnenat-Thalmann. A data-driven approach for real-time clothes simulation. *Computer Graphics Forum*, 24:173–183, 2005.

P. A. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2004.

Edilson de Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K. Hodgins. Stable spaces for real-time clothing. *ACM Trans. Graph.*, 29:106:1–106:9, July 2010. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1778765.1778843. URL `http://doi.acm.org/10.1145/1778765.1778843`.

G. Doretto and S. Soatto. Editable dynamic textures. In *IEEE Computer Vision and Pattern Recognition*, pages II: 137–142, 2003. URL

http://ieeexplore.ieee.org:80/xpls/abs_all.jsp?isNumber=27266&prod=CNF&arnumber=1211463&arSt=+137&ared=+142&arNumber=1211463.

Zdeněk Dostál and Joachim Schöberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Comput. Optim. Appl.*, 30(1):23–43, 2005. ISSN 0926-6003. doi: http://dx.doi.org/10.1007/s10589-005-4557-7.

Weinan E, Bjorn Engquist, Xiantao Li, Weiqing Ren, and Eric Vanden-Eijnden. Heterogeneous multiscale methods: A review. *Communications in Computational Physics*, 2 (3):367–450, 2007.

Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In ACM, editor, *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA*, pages 341–346, pub-ACM:adr, 2001. ACM Press. ISBN 1-58113-292-1.

Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, pages 1033–1038, 1999.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph*, 26(1), 2007. URL http://doi.acm.org/10.1145/1189762.1189766.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH*, pages 15–22, 2001. URL http://portal.acm.org/citation.cfm?id=383259.383260.

Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 904–909, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1186822.1073281. URL http://doi.acm.org/10.1145/1186822.1073281.

F. Feurtey. Simulating the collision avoidance behavior of pedestrians. Master's thesis, University of Tokyo, School of Engineering, February 2000.

P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Intl. J. on Robotics Research*, 17(7):760–772, 1998.

J. Fish. *Multiscale methods: bridging the scales in science and engineering.* Oxford University Press, 2009. ISBN 9780199233854.

Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH*, pages 23–30, 2001. URL `http://portal.acm.org/citation.cfm?id=383259.383261`.

Nick Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

Alain Fournier and William T. Reeves. A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1986. ACM Press. ISBN 0-89791-196-2. doi: http://doi.acm.org/10.1145/15922.15894.

Uriel Frisch. *Turbulence: The Legacy of A. N. Kolmogorov.* Cambridge University Press, 1995.

John J. Fruin. *Pedestrian planning and design.* Metropolitan Association of Urban Designers and Environmental Planners, 1971.

J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. *Proc. of ACM SIGGRAPH*, pages 29–38, 1999.

R. Gayle, W. Moss, M. C. Lin, and D. Manocha. Multi-robot coordination using generalized social potential fields. *Proc. IEEE Conf. Robotics and Automation*, 2009.

Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004)*, 23(3):463–468, 2004. URL `http://graphics.cs.berkeley.edu/papers/Goktekin-AMF-2004-08/`.

S. T. Greenwood and D. H. House. Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 287–296, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi: http://dx.doi.org/10.1145/1028523.1028562. URL `http://dx.doi.org/10.1145/1028523.1028562`.

S. Guy, J. Chhugani, C. Kim, N. Satish, M. C. Lin, D. Manocha, and P. Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

Jianwei Han, Kun Zhou, Li-Yi Wei, Minmin Gong, Hujun Bao, Xinming Zhang, and Baining Guo. Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput.*, 22(9):918–925, 2006. ISSN 0178-2789. doi: http://dx.doi.org/10.1007/s00371-006-0078-3.

Francis H. Harlow. The particle-in-cell method for numerical simulation of problems in fluid dynamics. In *Proc. Symp. Appl. Math.*, volume 15, 1963.

L. Heigeas, A. Luciani, J. Thollot, and N. Castagné. A physically-based particle model of emergent crowd behaviors. *Proc. Graphikon '03*, 2, 2003.

D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Sci.*, 39: 1–24, 2005.

Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282, May 1995. URL `http://link.aps.org/abstract/PRE/v51/p4282`.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In ACM, editor, *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA*, pages 327–340. ACM Press, 2001. ISBN 1-58113-292-1.

Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph.*, 24: 915–920, July 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073283. URL http://doi.acm.org/10.1145/1073204.1073283.

Roger L. Hughes. The flow of human crowds. *Annu. Rev. Fluid Mech.*, 35:169–182, 2003. doi: http://dx.doi.org/10.1146/annurev.fluid.35.101101.161136.

C. Josserand, P.-Y. Lagrée, and D. Lhuillier. Stationary shear flows of dense granular materials: a tentative continuum modelling. *The European Physical Journal E: Soft Matter and Biological Physics*, 14:127–135, 06 2004.

Michael Kass, Aaron Lefohn, and John Owens. Interactive depth of field. Technical Report 06-01, Pixar Animation Studios, 2006.

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, 27(5):164:1–164:11, 2008.

Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph.*, 26, July 2007a. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1276377.1276500. URL http://doi.acm.org/10.1145/1276377.1276500.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. Advections with significantly reduced dissipation and diffusion. *IEEE Trans. Vis. Comput. Graph*, 13(1): 135–144, 2007b. URL http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.3.

Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. A semi-Lagrangian CIP fluid solver without dimensional splitting. *Computer Graphics Forum*, 27:467–475(9), 2008a. doi: doi:10.1111/j.1467-8659.2008.01144.x. URL http://www.ingentaconnect.com/content/bpl/cgf/2008/00000027/00000002/art00034.

Doyub Kim, Oh-young Song, and Hyeong-Seok Ko. Stretching and wiggling liquids. *ACM Trans. Graph.*, 28:120:1–120:7, December 2009. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1618452.1618466. URL http://doi.acm.org/10.1145/1618452.1618466.

Janghee Kim, Deukhyun Cha, Byungjoon Chang, Bonki Koo, and Insung Ihm. Practical animation of turbulent splashing water. In Dieter Feneller and Stephen Spencer, editors, *Eurographics/SIGGRAPH Symposium on Computer Animation*. Eurographics Association, 2006.

Theodore Kim and Ming C. Lin. Stable advection-reaction-diffusion with arbitrary anisotropy. *Journal of Visualization and Computer Animation*, 18(4-5):329–338, 2007. URL http://dx.doi.org/10.1002/cav.187.

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. In *SIGGRAPH*, 2008b.

Vegard Kippe, Jørg Aarnes, and Knut-Andreas Lie. A comparison of multiscale methods for elliptic problems in porous media flow. *Computational Geosciences*, 12:377–398, 2008. ISSN 1420-0597. URL http://dx.doi.org/10.1007/s10596-007-9074-6. 10.1007/s10596-007-9074-6.

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 820–825, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179352.1141961. URL http://doi.acm.org/10.1145/1179352.1141961.

Joe Kniss and D. Hart. Volume effects: modeling smoke, fire, and clouds, 2004. Section from ACM SIGGRAPH 2004 courses, *Real-Time Volume Graphics*, http://www.cs.unm.edu/~jmk/sig04_modeling.ppt.

A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Proceedings of the USSR Academy of Sciences*, 30: 299–303, 1941.

Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23:559–568, August 2004. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1015706.1015760. URL `http://doi.acm.org/10.1145/1015706.1015760`.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21:473–482, July 2002. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/566654.566605. URL `http://doi.acm.org/10.1145/566654.566605`.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3): 277–286, July 2003. ISSN 0730-0301.

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics*, 24(3):795–802, July 2005. ISSN 0730-0301.

Vivek Kwatra, David Adalsteinsson, Theodore Kim, Nipun Kwatra, Mark Carlson, and Ming Lin. Texturing fluids. *To Appear in IEEE Transactions on Visualization and Computer Graphics*, 2007. DOI: 10.1109/TVCG.2007.1044.

T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein. Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution. *SIMULATION*, 81:339, 2005.

Horace Lamb. *Hydrodynamics*. University Press, 1993.

Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 541–548, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179352.1141921.

Toon Lenaerts and Philip Dutré. Mixing fluids and granular materials. *Computer Graphics Forum*, 28:213–218(6), 2009. doi: doi:10.1111/j.1467-8659.2009.01360.x. URL `http://www.ingentaconnect.com/content/bpl/cgf/2009/00000028/00000002/art00006`.

Xin Li and J. Michael Moshell. Modeling soil: realtime dynamic models for soil slippage and manipulation. In *SIGGRAPH '93*, pages 361–368. ACM, 1993. ISBN 0-89791-601-8. doi: http://doi.acm.org/10.1145/166117.166162.

M. C. Lin, S. Guy, R. Narain, J. Sewall, S. Patil, J. Chhugani, A. Golas, J. van den Berg, S. Curtis, D. Wilkie, P. Merrell, C. Kim, N. Satish, P. Dubey, and D. Manocha. Interactive modeling, simulation and control of large-scale crowds and traffic. *Proc. Workshop on Motion in Games (Springer-Verlag Lecture Notes in Computer Science Series)*, 2009.

Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35:2006, 2005.

C. Loscos, D. Marchal, and A. Meyer. Intuitive crowd behaviour in dense urban environments using local laws. In *Theory and Practice of Computer Graphics*, pages 122–129, 2003.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

A. Luciani, A. Habibi, and E. Manzotti. A multi-scale physical model of granular materials. In *Proceedings of Graphics Interface*, 1995.

Jacques Magnaudet. High-Reynolds-number turbulence in a shear-free boundary layer: revisiting the HuntâĂŞGraham theory. *Journal of Fluid Mechanics*, 484:167–196, 2003.

Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. Image-based brdf measurement including human skin. In *Proceedings of 10th Eurographics Workshop on Rendering*, pages 139–152, 1999.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22:759–769, July 2003. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/882262.882343. URL `http://doi.acm.org/10.1145/882262.882343`.

W. D. McComb. *The Physics of Fluid Turbulence*. Oxford University Press, 1990.

Tom Mertens, Jan Kautz, Jiawen Chen, Philippe Bekaert, and Frédo Durand. Texture transfer using geometry correlation. In Tomas Akenine-Möller and Wolfgang Heidrich, editors, *Proceedings of Eurographics Symposium on Rendering 2006*. Eurographics Association, 2006.

Ronald A. Metoyer and Jessica K. Hodgins. Reactive pedestrian path following from examples. In *Proc. 16th Int. Conf. Computer Animation and Social Agents*, page 149, 2003. ISBN 0-7695-1934-2.

Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989. URL `citeseer.ist.psu.edu/miller89globular.html`.

G. W. Milton. *The theory of composites*, volume 6 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2002.

A. S. Monin and A. M. Yaglom. *Statistical Fluid Mechanics: Mechanics of Turbulence*, volume 1. MIT Press, 1971.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph.*, 28:38:1–38:8,

July 2009. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1531326.1531344. URL http://doi.acm.org/10.1145/1531326.1531344.

Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In D. Breen and M. Lin, editors, *Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 154–159, San Diego, California, 2003. Eurographics Association. URL http://www.eg.org/EG/DL/WS/SCA03/154-159.pdf.

S. R. Musse and D. Thalmann. A model of human crowd behavior: Group interrelationship and collision detection analysis. *Computer Animation and Simulation*, pages 39–51, 1997.

Rahul Narain, Vivek Kwatra, Huai-Ping Lee, Theodore Kim, Mark Carlson, and Ming C. Lin. Feature-guided dynamic texture synthesis on continuous flows. In Jan Kautz and Sumanta Pattanaik, editors, *Rendering Techniques 2007 (Proc. Eurographics Symposium on Rendering)*, pages 361–370, 2007.

Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 27(5), 2008.

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 28 (5), 2009.

Rahul Narain, Abhinav Golas, and Ming C. Lin. Free-flowing granular materials with two-way solid coupling. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 29 (6), 2010.

Fabrice Neyret. Advected textures. In D. Breen and M. Lin, editors, *Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 147–153, San Diego,

California, 2003. Eurographics Association. URL http://www.eg.org/EG/DL/WS/SCA03/147-153.pdf.

Koichi Onoue and Tomoyuki Nishita. Virtual sandbox. In *Pacific Conference on Computer Graphics and Applications*, page 252. IEEE Computer Society, 2003. ISBN 0-7695-2028-6. doi: http://doi.ieeecomputersociety.org/10.1109/PCCGA.2003.1238267.

Sebastien Paris, Julien Pettre, and Stephane Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674, September 2007. ISSN 0167-7055. doi: 10.1111/j.1467-8659.2007.01090.x. URL http://dx.doi.org/10.1111/j.1467-8659.2007.01090.x.

Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In Demetri Terzopoulos and Victor Zordan, editors, *ACM SIGGRAPH /Eurographics Symposium on Computer Animation*, pages 261–270, Los Angeles, California, 2005. Eurographics Association. ISBN 1-59593-198-8. doi: 10.2312/SCA/SCA05/261-270. URL http://www.eg.org/EG/DL/WS/SCA/SCA05/261-270.pdf.

Grigorios A. Pavliotis and Andrew M. Stuart. *Multiscale Methods: Averaging and Homogenization*. Texts in Applied Mathematics. Springer New York, 2008.

N. Pelechano, K. O'Brien, B. Silverman, and N. Badler. Crowd simulation incorporating agent psychological models, roles and communication. *First Intl. Workshop on Crowd Simulation*, 2005.

N. Pelechano, J. M. Allbeck, and N. I. Badler. *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers, 2008.

K. Perlin. An image synthesizer. *Computer Graphics*, 19(3):287–296, July 1985.

Ken Perlin. Improving noise. *ACM Transactions on Graphics*, 21(3):681–682, July 2002. ISSN 0730-0301.

Ken Perlin and Fabrice Neyret. Flow noise. In *Siggraph Technical Sketches and Applications*, page 187, Aug 2001. URL http://evasion.imag.fr/Publications/2001/PN01.

Julien Pettré, Marcelo Kallmann, and Ming C. Lin. Motion planning and autonomy for virtual humans. In *ACM SIGGRAPH 2008 classes*, pages 1–31, 2008. doi: http://doi.acm.org/10.1145/1401132.1401193.

Tobias Pfaff, Nils Thürey, Andrew Selle, , and Markus Gross. Synthetic Turbulence using Artificial Boundary Layers. *ACM SIGGRAPH Asia 2009 Papers*, 28,5:10, December 2009.

Tobias Pfaff, Nils Thuerey, Jonathan Cohen, Sarah Tariq, and Markus Gross. Scalable fluid simulation using anisotropic turbulence particles. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pages 174:1–174:8, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0439-9. doi: http://doi.acm.org/10.1145/1866158.1866196. URL http://doi.acm.org/10.1145/1866158.1866196.

Marta Pla-Castells, Ignacio García-Fernández, and Rafael J. Martínez. Interactive terrain simulation and force distribution models in sand piles. In Samira El Yacoubi, Bastien Chopard, and Stefania Bandini, editors, *Cellular Automata*, volume 4173 of *Lecture Notes in Computer Science*, pages 392–401. Springer, 2006. ISBN 3-540-40929-7.

F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualization: Feature extraction and tracking. *Computer Graphics Forum*, 22(4): 775–792, 2003. URL http://www.vris.at/ar3/pr2/star/.

Simon Premoze, Tolga Tasdizen, James Bigler, Aaron E. Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Comput. Graph. Forum*, 22(3):401–410, 2003. URL http://diglib.eg.org/EG/CGF/volume22/issue3/cgf_687.html.

M. Quecedo, M. Pastor, M. I. Herreros, and J. A. Fernández Merodo. Numerical modelling of the propagation of fast landslides using the finite element method. *International Journal for Numerical Methods in Engineering*, 59(6):755–794, 2004.

W. Ren and W. E. Heterogeneous multiscale method for the modeling of complex fluids and microfluidics. *Journal of Computational Physics*, 204(1):1–26, 2005.

C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH*, 21:25–34, 1987.

C. W. Reynolds. Steering behaviors for autonomous characters. *Game Developers Conference*, 1999, 1999.

Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26, July 2007. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1276377.1276510. URL `http://doi.acm.org/10.1145/1276377.1276510`.

Hagit Schechter and Robert Bridson. Evolving sub-grid turbulence for smoke animation. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2008.

Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5. doi: http://doi.acm.org/10.1145/344779.345012.

M. Schreckenberg and S. D. Sharma. *Pedestrian and Evacuation Dynamics*. Springer, 2001.

B. G. Schunck and B. K. P. Horn. Constraints on optical flow computation. In *Pattern Recognition and Image Processing*, pages 205–210, 1981.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24(3):910–914, July 2005. ISSN 0730-0301.

Andrew Selle, Ronald Fedkiw, ByungMoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable MacCormack method. *Journal of Scientific Computing*, 2008. (in press).

Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graph. Models*, 69(5-6): 246–274, 2007. ISSN 1524-0703. doi: http://dx.doi.org/10.1016/j.gmod.2007.09.001.

Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

Mikio Shinya and Alain Fournier. Stochastic motion - motion under the influence of wind. *Computer Graphics Forum*, 11(3):119–128, September 1992. EG92: Cambridge, UK., Editors: A. Kilgour and L. Kjelldahl.

J.C. Simo and T.J.R. Hughes. *Computational Inelasticity*. Springer, 1998.

Karl Sims. Particle animation and rendering using data parallel computation. In *Computer Graphics, Proceedings of Siggraph*, volume 24, pages 405–413. ACM, August 1990.

Jos Stam. Stable fluids. In *SIGGRAPH*, pages 121–128, 1999. URL `http://doi.acm.org/10.1145/311535.311548`.

Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH*, pages 369–376. ACM, 1993. ISBN 0-89791-601-8. URL `http://doi.acm.org/10.1145/166117.166163`.

A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha. Real-time path planning in dynamic virtual environments using multi-agent navigation graphs. *IEEE Trans. Visualization and Computer Graphics*, 14(3):526–538, 2008.

Avneesh Sud, Russell Gayle, Erik Andersen, Stephen Guy, Ming Lin, and Dinesh Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *Proc. ACM Symp. Virtual Reality Software and Technology*, pages 99–106, 2007. ISBN 978-1-59593-863-3. doi: http://doi.acm.org/10.1145/1315184.1315201.

Y. Sugiyama, A. Nakayama, and K. Hasebe. 2-dimensional optimal velocity models for granular flows. In *Pedestrian and Evacuation Dynamics*, pages 155–160, 2001.

Robert W. Sumner, James F. O'Brien, and Jessica K. Hodgins. Animating sand, mud, and snow. *Computer Graphics Forum*, 18(1):17–26, 1999.

M. Sung, M. Gleicher, and S. Chenney. Scalable behaviors for crowd simulation. *Computer Graphics Forum*, 23(3 (Sept)):519–528, 2004.

T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki. Realistic animation of fluid with splash and foam. *Computer Graphics Forum*, 22(3): 391–400, 2003. doi: http://dx.doi.org/10.1111/1467-8659.00686.

D. Thalmann, C. O'Sullivan, P. Ciechomski, and S. Dobbyn. *Populating Virtual Environments with Crowds.* Eurographics 2006 Tutorial Notes, 2006.

Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

A. Treuille, S. Cooper, and Z. Popovic. Continuum crowds. *ACM Trans. Graph.*, 25(3): 1160–1168, 2006a.

Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 826–834, New York, NY, USA, 2006b. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179352.1141962. URL `http://doi.acm.org/10.1145/1179352.1141962`.

Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, and Ming C. Lin. Interactive navigation of individual agents in crowded environments. *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games*, pages 139–147, 2008.

Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. *Proceedings of the International Symposium on Robotics Research*, 2009.

Gregory J. Ward. Measuring and modeling anisotropic reflection. *SIGGRAPH Comput. Graph.*, 26:265–272, July 1992. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/142920.134078. URL `http://doi.acm.org/10.1145/142920.134078`.

Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 479–488. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000. URL `http://visinfo.zib.de/EVlib/Show?EVL-2000-87`.

Jakub Wejchert and David Haumann. Animation aerodynamics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 19–22, New York, NY, USA, 1991. ACM. ISBN 0-89791-436-8. doi: http://doi.acm.org/10.1145/122718.122719. URL `http://doi.acm.org/10.1145/122718.122719`.

Martin Wicke, Matt Stanton, and Adrien Treuille. Modular bases for fluid dynamics. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 39:1–39:8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: http://doi.acm.org/10.1145/1576246.1531345. URL `http://doi.acm.org/10.1145/1576246.1531345`.

Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 364–367, New York, NY, USA, 2004. ACM Press. doi: http://doi.acm.org/10.1145/1186562.1015730.

H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. C. Lin. Composite agents. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008.

Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics*, 22(3):295–302, July 2003. ISSN 0730-0301.

Yongning Zhu and Robert Bridson. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, pages 965–972. ACM, 2005. doi: http://doi.acm.org/10.1145/1186822.1073298.