Diss. ETH No. 20563

Detail Enhancement for Fluid Simulations using Turbulence Modeling

A dissertation submitted to
ETH Zürich

for the Degree of
Doctor of Sciences

presented by
Tobias Pfaff
Dipl. Phys., Universität Konstanz, Germany
born 3. October 1980
citizen of Germany

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. James O'Brien, co-examiner
Dr. Nils Thürey, co-examiner
Prof. Dr. Ronald Peikert, co-examiner

2012

Abstract

The complex and exciting appearance of the natural phenomena of smoke, water and fire make them powerful tools to convey realism and create visually thrilling settings in virtual scenarios and movies. Their characteristic look is determined by the chaotic nature of turbulent flows. Unfortunately, turbulence also makes it hard to reproduce these flows in numerical simulations, as it induces an immense amount of detailed motion which needs to be stored and processed.

Instead of direct simulation, we study and *model* turbulence production and transition processes in suitable *higher level representations*. This allows the generation of *synthetic turbulence* at arbitrary resolution, thereby producing very detailed results at a fraction of the cost of a full simulation. This thesis explores new ways to make these powerful tools available for a wide range of flow phenomena.

The first part of this thesis thesis presents a particle-based turbulence representation and prediction model that is suitable for *real-time simulation* of highly-detailed turbulent flows. We use a complete energy transfer model to predict turbulence intensity, and curl-noise texture based synthesis to augment a base simulation with synthetic detail. To correctly reproduce the anisotropic turbulence generation, we extend our predictor and synthesis step for planar anisotropy based on Reynold stress transport theory. Relying on the turbulence model for complex dynamics and eliminating feedback to the main solver enables a design that scales very well on parallel hardware.

In the second part, we investigate a *vorticity-based turbulence representation*. This representation is suitable for modeling the breakdown of coherent structures into turbulence, which is hard to achieve in a curl-noise texture formulation. To predict turbulence generation, we precompute and track the boundary layer around flow obstacles, and estimate flow instabilities using turbulence modeling. This results in turbulence seeded with full anisotropy information which seamlessly integrates in the main flow. Our precomputation allows us to predict turbulence seeding from even complex obstacles below simulation resolution.

The third part of the thesis focuses on the problem of simulating detailed large-scale buoyant plumes. We propose a method that *only operates on the plume's interface surface* without the need of volumetric computations. We directly model the baroclinity-driven vortex sheets at the interface. This makes it possible to reproduce

the characteristic cloud billowing effect, which is a transition effect therefore cannot be represented by turbulence synthesis. To make this approach orthogonal to other turbulent methods, we introduce a turbulence predictor which discriminates obstacle-induced and free-stream baroclinic turbulence. We demonstrate that this method can be easily combined with the bulk turbulence approach in the first part of the thesis. By reducing the dimensionality of the problem, and the introduction of a local evaluation scheme, this method is very efficient for large-scale phenomena.

Zusammenfassung

Die Natuerphänomene Rauch, Wasser und Feuer sind wegen ihrer Komplexität und ihrem faszinierenden Erscheinungsbild wirksame Mittel, um in Filmszenen Authentizität und Spannung zu vermitteln. Ihr charakteristisches Aussehen erhalten diese Phänomene durch die chaotische Dynamik turbulenter Flüssigkeiten. Diese Turbulenzdynamik erzeugt jedoch eine ganze Kaskade an kleinskaligen Bewegungen in der Flüssigkeit, die gespeichert und berechnet werden müssen. Dies macht es schwierig, turbulenten Fluss numerisch zu simulieren.

Anstelle einer direkten Simulation, analysieren und modellieren wir die Entstehungs- und Übergangsprozesse von Turbulenzen daher in einer geeigneten übergeordneten Darstellung. Dies macht es möglich, Turbulenzen in beliebigen Auflösung zu synthetisieren, und damit detaillierte Flussfelder zu einem Bruchteil der Kosten einer vollen Simulation zu erhalten. Diese Arbeit erschliesst neue Ansätze, die mächtigen Instrumente der Turbulenzmodellierung und Synthese für eine Vielzahl Flussphänomenen verfügbar zu machen.

Der erste Teil dieser Arbeit stellt eine partikelbasierte Turbulenzdarstellung und ein Vorhersagemodell vor, die die Simulation von detaillierten turbulenten Flüssen in Echtzeit ermöglichen. Wir verwenden ein komplettes Energietransfermodell zur Vorhersage der Turbulenzintensität und einen Synthesealgorithmus mit Curl-Noise Texturen, um eine Basissimulation mit synthetischen Details zu erweitern. Um die anisotrope Turbulenzproduktion korrekt wiedergeben zu können, wird die Turbulenzvorhersage und Synthese mit einem planeren Anisotropiemodell basierend auf einem Reynolds-Spannungstransportmodell erweitert. In unserem Modell wird die komplexe Dynamik der kleinen Skalen zu grossen Teilen vom Turbulenzmodell übernommen und eine Rückkopplung der synthetischen Details in die Basissimulation vermieden. Dies ermglicht ein Design, das sehr gut auf paralleler Hardware skaliert.

Im zweiten Teil der Arbeit untersuchen wir die Repräsentation von Turbulenz durch die Wirbelstromstärke. Im Gegensatz zu Curl-Noise Texturen ist diese Darstellung sehr gut geeignet, um den Zerfall von kohärenten Strukturen zu Turbulenz zu beschreiben. Zur Vorhersage der Turbulenzentstehung ermitteln wir die Grenzschicht von Hindernissen im Fluss in einem Vorberechnungsschritt, und bestimmen Instabilitäten im Fluss mittels Turbulenzmodellierung. Die syn-

thetischen Turbulenzen besitzen somit vollständige Anisotropieinformation, und integrieren sich nahtlos in den Fluss der Basissimulation. Durch Vorberechnung können wir sogar die Erzeugung von Turbulenzen durch Hindernisse unter der Auflösungsgrenze der Simulation korrekt vorhersagen.

Der dritte Teil der Arbeit konzentriert sich auf die Simulation von grossskaligen, aufsteigenden Rauchschwaden. Wir stellen eine Methode vor, die direkt auf der Grenzfläche der Rauchwolke operiert, und ohne Berechnungen im inneren Volumen auskommt. Wir modellieren die durch Baroklinität hervorgerufene Wirbelschicht an der Grenzfläche, und erhalten dadurch die charakteristischen Oberflächenformationen von Quellwolken, die mit Turbulenzsynthese schwer zu reproduzieren sind. Wir führen zusätzlich ein Vorhersagemodell ein, das zwischen Turbulenzentstehung im freien Fluss und Entstehung an Flusshindernissen unterscheidet. Dies macht unser Modell kompatibel zu anderen volumetrischen Turbulenzmodellen, was wir anhand des Modells im ersten Teil der Arbeit demonstrieren. Die Reduzierung der Dimensionalität des Problems und die Einführung eines lokalen Auswerteschemas machen diese Methode sehr effizient für die Simulation von grossskaligen Phänomenen.

Acknowledgments / Danksagung

My sincere thanks go to my advisor Prof. Markus Gross. I have been very fortunate to have an advisor who gave me the freedom to explore my own ideas, and at the same time guidance when I was 'lost in research'. His continuing support for my work, and his scientific intuition were invaluable to make this thesis possible. I am also very thankful to Dr. Nils Thürey – I was in the lucky position to learn from one of the best in the field of fluid simulation. Nils introduced me into the wondrous world of fluid simulation in Graphics during my first years, helped me through the hurdles of writing my first research paper and stayed a close collaborator for many recent projects. I am very grateful to have Nils as a supervisor, collaborator and last but not least as a friend. I would also like to thank Prof. James O'Brien for his openness and interest in my ideas, and the fruitful discussions at Berkeley. I am very happy to have him as an examiner for my thesis. I'm grateful to Prof. Ronald Peikert, who jumped in at the last minute to save this thesis from the secret, evil directive on videoconferencing.

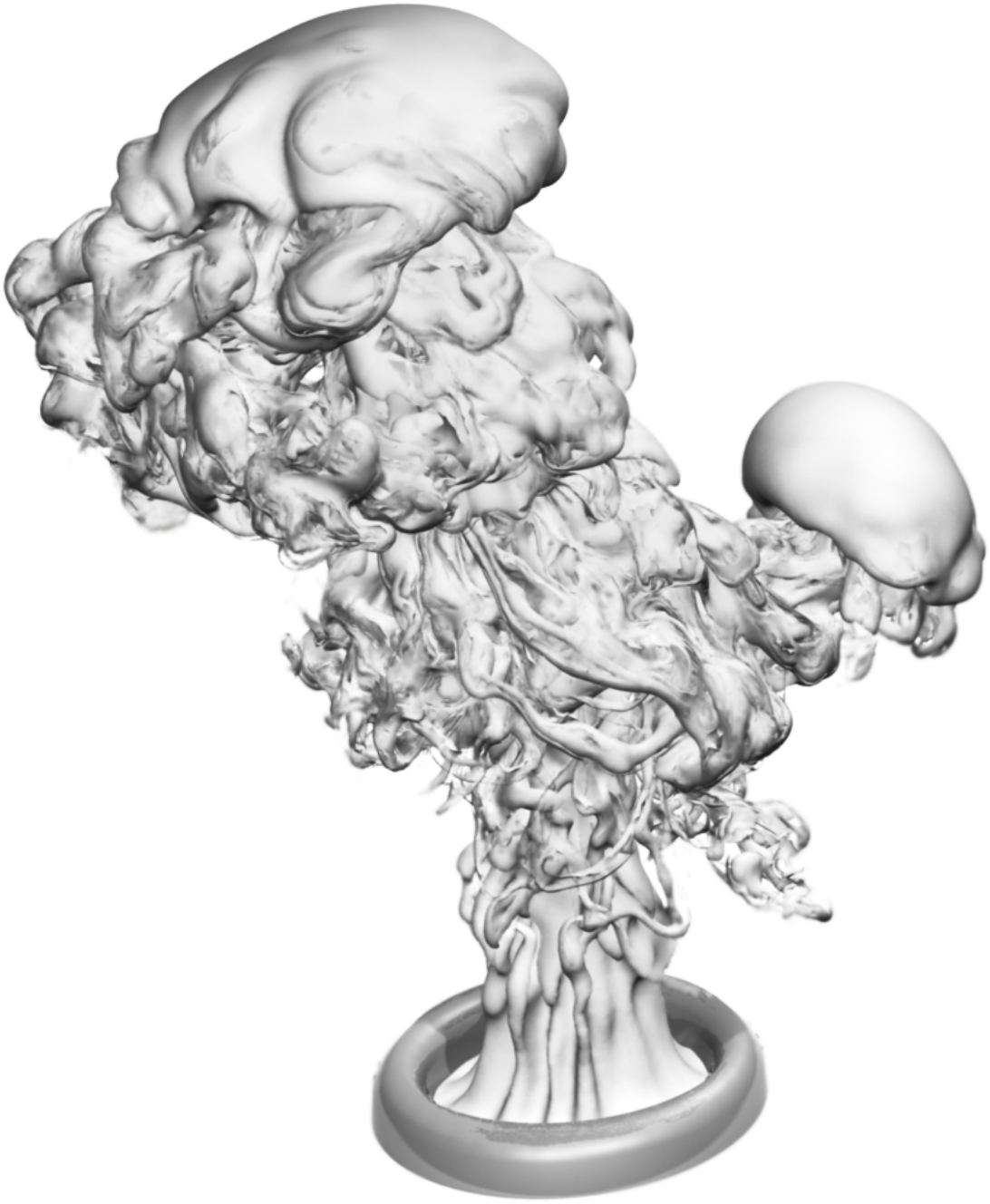
Many thanks go to my collaborators Jon Cohen, Sarah Tariq and Andrew Selle who contributed in many ways to the research work in this thesis. It was a pleasure to work with them and I am thankful for their effort and commitment to the realized projects. I would also like to thank Matthias Hölling from ETH and the nVidia tech team for making the world's fastest exchange of source code across difficult copyright territory possible, which allowed us to meet the deadline. Special thanks go to Theodore Kim, Chris Wojtan, Sebastian Martin and Cengiz Öztireli for their insightful thoughts and inspiring discussions about various topics in simulation.

Furthermore, I also want to thank my former office mates in the glorious hippo office, Thomas Oskam and Marcel Germann, together with all the good people at CGL, IGL, AGG and DRZ for the great time at ETH. No matter what time of day or night, someone was always around for a coffee and a nice chat. Thanks also to my improv people, who made sure I get out of the lab once in a while even during stressful times, and kept my general sanity up.

This work has been made possible by the ETH grant TH-23 07-3.

Ich sage 'Danke' an alle meine Freunde und meine Familie für ihre Unterstützung und ihr Verständnis in diesen vier Jahren. Schliesslich will ich noch meiner Freundin Sonja danken, die mit mir durch diese manchmal schöne, manchmal auch schwere Zeit gegangen ist und mich immer mit einem Lächeln im Gesicht und einem Plüschrochen auf der Schulter unterstützt hat.

Contents



"When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first."

Werner Heisenberg, according to a dubious source

CHAPTER 1

Introduction

The natural phenomena of smoke, water and fire are fascinating to watch, be it in nature or in movies. What makes them so interesting is the incredible amount of detail they exhibit. In smoke plumes from volcanoes or large explosions, complex surface structures at scales from the millimeter range to several meters are visible. The characteristic look of fire is determined by the chaotic movements of the flames, driven by small whirls. And perhaps the most classical example, white-water rapids only look exciting due to their intricate dynamics, from little splashes to big vortices. Their complex and exciting appearance make these effects powerful tools to convey realism and to create visually thrilling settings in movies and computer games. Numerical simulations used in these areas therefore need to be able to reproduce such dynamics.

Most of the chaotic behavior of fluids can be ascribed to their turbulent nature. At high velocities and low viscosity values, fluid flows quickly become instable, and a whole cascade of vortical movement forms, turning laminar into turbulent flow. The fact that small instabilities can lead to very different large-scale dynamics makes turbulent flows hard to control and predict in simulations. This is a fundamental issue for most areas which deal with the simulation of fluids, such as engineering or meteorology, and the reason why fluid simulation is considered a difficult problem. For simulations in Computer Graphics, another problem arises. As the fluid motion is to be

visualized, the scene has to be represented from the smallest details to the largest scales. This requires a very high simulation resolution, and therefore computation time, since the defining equations for fluid motion scale badly with increasing resolution. At the same time, fewer computational resources are available compared to applications in other sciences, as fast turnaround times in the case of movies, or even real-time response in the case of computer games is required.

Although current movies present convincing fluid animation scenes, this problem is far from being solved. To obtain such visual quality, animators have to operate under a number of restrictions and rely on costly manual operations. Fluid effects shots are often very short and heavily edited using artist-defined particles systems and additional rendering layers. For large simulations, the artistic process becomes even more involved due to long turnaround times. Techniques to increase the detail level without the associated cost of a full simulation would remove some of these restrictions, providing more stylistic freedom to animators and opening up new possibility for fluid simulations in e.g. interactive applications.

In Computer Graphics research, many approaches have been developed to alleviate the problems mentioned above. A large part of previous work focuses on reducing *numerical viscosity*, which dampens out small-scale turbulence. This can be performed by e.g. improving accuracy of the advection [Selle et al., 2008] or using energy-conserving integration schemes [Mullen et al., 2009]. While such methods have been applied with success, the fundamental issues remain as the potential detail scale is still limited to the simulation resolution.

In this thesis, we will take a different approach to the resolution problem. In our methods, we detach turbulent details from the main fluid flow using Reynolds averaging [Pope, 2000]. Instead of a direct simulation of the motion detail using the equations of fluid dynamics, we aim to understand, analyze and model the processes leading to the creation and evolution of turbulence. Reasoning about turbulence in a high-level manner is very advantageous for a number of reasons. Most importantly, we are able to store and process data about turbulence in a compact and efficient manner. Based on this information, we are able to synthesize turbulence at arbitrary resolutions at a fraction of the cost of a full simulation. But the separation of turbulence also allows artists to manipulate turbulence in an intuitive manner without disturbing the flow, it makes it possible to augment existing flow data in a post-production step, and opens up new possibility for LOD adaptivity. This procedure however only works for turbulent processes which are understood, correctly modeled and can be represented and synthesized in a meaningful

manner. Otherwise, the synthesis will produce visual artifacts or unrealistic flow behavior. In this thesis, we explore new ways for representing turbulence and modeling flow processes, in order to make the powerful tools of detail separation available for a wide range of flow phenomena.

1.1 Thesis Overview

In the course of this thesis, we will address the topic of detail enhancement and turbulence from various angles, for different applications. The research performed is summarized below, classified in advancements in modeling and synthesis.

Representation and Synthesis We first investigate representations for turbulence that are both efficient and powerful. Especially for performance-sensitive applications, turbulent kinetic energy (TKE) based models are an evident choice, as their behavior is well-studied in Computational Fluid Dynamics (CFD), and they work well in combination with fast frequency-matched curl noise synthesis [Kim et al., 2008b]. However, unlike Kim et al. we choose Lagrangian particles instead of a volumetric grid to store the turbulence information. We unify the representation of turbulence and smoke density, which allows us to represent sub-grid detail exactly where it is needed for rendering. This representation also enables a solver design which scales very well on parallel hardware. On the other hand, curl noise synthesis and TKE-based models are not suitable for representing the important class of turbulence production effects, as these processes are highly anisotropic. To improve the generality of this approach, we also predict and track anisotropy, and develop a modified synthesis algorithm to generate the subclass of planar anisotropic turbulence which is most often encountered in production processes.

Even with anisotropy extensions, curl-noise synthesis has its limits. The breakdown of coherent structures into turbulence, and the alignment of turbulence features to the base flow can hardly be mapped to such a representation. We therefore propose vorticity-based approaches to simulate complex flow processes. As a general turbulence representation, we study vortex particles [Selle et al., 2005] which allow coherent structures and complex anisotropy modeling. Compared to the original method, we synthesize the turbulence using Biot-Savart integration instead of forcing, which decouples turbulence further from the underlying simulation and allows sub-grid synthesis. As

energy transfer via vortex stretching is hard to stabilize in a sparse vortex particle setting, we calculate energy transfer based on the K41 spectrum.

Certain types of flows, such as buoyant plumes, are completely determined by the vorticity dynamics on the plume interface. By tracking this plume interface with a Lagrangian surface mesh, and evaluating the vorticity-induced motion equations on this surface, it is therefore possible to obtain the flow dynamics including details on mesh resolution level without any volumetric computations [Stock et al., 2008]. The velocity integration is however of quadratic complexity, and therefore scales badly. Also, certain effects such as the interaction with flow obstacles are hard to model in this representation. We introduce a local evaluation scheme which reduces the numerical complexity of this operation by coupling it to a low-resolution Navier-Stokes solver in a manner similar to turbulence separation. This also allows us to relax the constraint of purely buoyancy-driven dynamics, such that complex base flows with obstacles interaction can be used.

Modeling Turbulence models predict the spatial distribution and evolution of turbulence properties such as the TKE. Direct predictor methods use small eddies in the base flow as turbulence indicators [Kim et al., 2008b; Fedkiw et al., 2001], which breaks down if turbulence production happens on scales below the grid resolution. Schechter et al. [2008] on the other hand use a simple energy transport model for prediction. As this model is incomplete, it relies on a uniform turbulent viscosity model, which is only valid for a narrow set of flows with fully-developed turbulence. We therefore introduce a modified version of the k–ε model to Graphics, which is a complete, general-purpose model which predicts correct production terms for a wide range of settings.

While TKE models are able to provide valid predictions for turbulence generation and transition, within certain boundaries, the TKE representation does not contain the relevant information to synthesize the respective transition motion. Vorticity-based turbulence representations, on the other hand, are better suited for this task, but there are no standard models for turbulence modeling and prediction in this formulation. We therefore develop a hybrid model for modeling turbulence generation at obstacles in vorticity space, which uses the reliable TKE methodology for prediction of flow instability, while tracking anisotropy and vortex strength in a vorticity formulation. This enables the accurate simulation of vortex shedding from obstacles with complex geometry.

Finally, we also study baroclinic instabilities which give rise to the charac-

teristic 'billowing' observed in smoke plumes stemming from explosions or volcanic activity. As this a transition effect, it cannot be reproduced using TKE turbulence models. We find that this effect is best modeled by solving baroclinic vorticity dynamics on the plume interface. Synthesis is performed using Biot-Savart integration directly on the surface mesh, yielding baroclinic sub-grid detail efficiently. To be able to use this model in a general case, which also includes turbulence seeded from obstacle interaction, we develop a turbulence predictor which discriminates obstacle and baroclinic turbulence. The obstacle-induced component can then be handled by a TKE turbulence model as described above.

1.2 Contributions

The thesis makes the following principal contributions:

- A *robust and scalable particle based TKE turbulence model* that is designed to work without particle-particle interaction, and is therefore suitable for massively parallel computations. (Chapter 4)
- A synthesis and prediction model for *anisotropic turbulence* in TKE formulation which efficiently captures directional vortices, thereby realistically integrating the turbulence into the base flow. (Chapter 4)
- A *vorticity-based predictor* for the generation of turbulence at flow obstacles. By precomputing boundary layer vorticity, and using a TKE model to predict instabilities we can correctly predict turbulent vortex shedding. (Chapter 5)
- A *modified vortex particle* representation which uses direct velocity synthesis to decouple from the base flow, and uses K41-based energy dynamics to avoids vortex-stretching instabilities.(Chapter 5)
- A *baroclinity model* based on vortex sheets, which is able to reproduce the billowing effect. By evaluating the model only the interface surface and using a local evaluation scheme, we obtain a high level of detail very efficiently. (Chapter 6)
- A *selective turbulence predictor* which separates turbulence energy production from free-stream baroclinity and obstacle interaction, and makes it possible to orthogonally combine the vortex sheet-based baroclinity model with a TKE model without overlap. (Chapter 6)

1.3 Thesis outline

This thesis is organized in the following manner: **Chapter 2** reviews the related work in the fields of turbulence modeling and vortex methods in

Computer Graphics and CFD. As this thesis builds upon classical turbulence modeling theory, and makes use of different velocity and vorticity representations to describe fluid motion, these concepts will be introduced in **Chapter 3**. **Chapter 4** presents our general-purpose anisotropic TKE model, and a particle-based synthesis method. We will demonstrate the real-time simulation of detailed turbulent flows using these methods. In **Chapter 5**, the development of turbulence in the wake of flow obstacles is studied. We will present a predictor which directly models this process using artificial boundary layers and TKE-based instability prediction, and a turbulence synthesis step which uses a modified vortex particle representation. The simulation of detailed large-scale buoyant plumes by solving vortex sheet equations on an interface surface mesh is described in **Chapter 6**. **Chapter 7** concludes the thesis by providing an assessment of the techniques presented in this thesis and suggesting potential further research topics. The source code to many of the methods discussed here is publicly available in the framework *Mantaflow* (**Appendix A.3**), which was developed during the course of this thesis.

1.4 Publications

The methods presented in this thesis have been published in the following peer-reviewed journals:

- T.PFAFF, N. THUEREY and M. GROSS. Lagrangian Vortex Sheets for Animating Fluids. In *Proceedings of ACM SIGGRAPH (Los Angeles, USA, August 5-9, 2012), ACM Transaction on Graphics, vol. 31, no. 4, pp. 112:1–112:8*.

  This paper describes our vortex sheet model for efficiently simulating detailed buoyant smoke plumes.

- T. PFAFF, N. THUEREY, J. COHEN, S. TARIQ and M. GROSS. Scalable Fluid Simulation using Anisotropic Turbulence Particles. In *Proceedings of ACM SIGGRAPH Asia (Seoul, Korea, December 15-18, 2010), ACM Transaction on Graphics, vol. 29, no. 5, pp. 174:1–174:8.*

  In this work, we present an anisotropic TKE turbulence model and particle-based synthesis method suitable for real-time fluid simulation.

- T. PFAFF, N. THUEREY, A. SELLE and M. GROSS. Synthetic Turbulence using Artificial Boundary Layers. In *Proceedings of ACM SIGGRAPH Asia (Yokohama, Japan, December 16-19, 2009), ACM Transactions on Graphics, vol. 28, no.5, pp. 121:1–121:10.*

  This paper introduces a vorticity-based predictor for obstacle-induced turbulence, based on boundary layer modeling and TKE instability prediction.

CHAPTER 2

Related Work

Methods for modeling and simulating fluid system have a long tradition in the fields of engineering and physics, and have become vital tools in Computer Graphics, too. The requirements for applications in Computer Graphics are however very different from their counterparts in CFD. Therefore, while similar in theoretical background, methods in Graphics often approach the problems at hand from a different angle, and much is to be learned from studying both sides. In this chapter we will summarize the history and recent works from both Computer Graphics and CFD, focusing on methods for simulating turbulent fluid systems. A good overview can also be found in the textbooks by Wilcox [1993] and Pope [2000] for turbulence theory, and the recapitulation of fluid simulation methods in Computer Graphics by Bridson [2008].

2.1 History

Fluid dynamics are described by the Navier-Stokes equations, a set of partial differential equations. To solve these equations numerically, several discretization schemes have been proposed. Most applications in Engineering base on either finite elements (FEM) [Oden and Wellford, 1972] or finite difference (FDM) / finite volume (FVM) [Hsu, 1981] discretizations. Finite

difference methods operate on structured grids, while finite element methods evaluate base functions on irregular meshes. The latter allows to focus resolution on critical regions if known in advance, but require a separate meshing step and have a larger computational overhead. Marker-and-Cell (MAC) discretizations [Harlow and Welch, 1966] extend the FDM approach by placing velocity information on the cell faces instead of centers, which increases precision when calculating derivatives.

Although most methods for numerical simulation of fluid systems use a direct discretization of the Navier-Stokes equations, different principles have been proposed in literature. The fully Lagrangian Smoothed Particle Hydrodynamics (SPH) models fluid dynamics by the interaction of particles with a compressible kernel [Gingold and Monaghan, 1977]. This method has become popular in Computer Graphics for free-surface problems [Müller et al., 2005]. It does however require small timesteps for stability in complex scenarios. Lattice Boltzmann methods on the other hand base on Boltzmann gas dynamics [Hardy et al., 1976], and directly model the flux and collision of fluxes on a regular mesh. While not common in Graphics, they have successfully been used for the simulation of liquids [Thuerey et al., 2006].

2.2 Fluid Simulation in Computer Graphics

Fluid simulation in Computer Graphics was popularized by Stam [1999], who introduced the combination of semi-Lagrangian advection with first order pressure projection using a MAC discretization. This unconditionally stable, albeit very dissipative type of solver is the most commonly used simulation technique in Computer Graphics and it will serve as reference and base solver in this thesis as well.

Over the years, many extensions of this basic solver have been made. This includes for example methods to simulate liquids [Enright et al., 2002a], bubble flows [Hong and Kim, 2003], viscoelastic fluids [Goktekin et al., 2004], or interactions with rigid bodies [Carlson et al., 2004]. Possibly the biggest challenge for fluid simulation for Computer Graphics, however, is to represent highly detailed flows efficiently. This problem is two-fold, and due to the dissipative nature of stable semi-Lagrangian advection, and the more fundamental issue of the memory and computation costs involved in high-resolution simulations. Below, recent methods to alleviate issues concerning dissipation, grid resolution and efficiency are summarized.

2.2.1 Low-dissipative Methods

The first part of the problem in preserving detail is the inherent damping of turbulence detail due to numerical dissipation. A popular approach to alleviate this problem is the use of higher order advection schemes. Back and Forth Error Correction [Kim et al., 2005], MacCormack advection [Selle et al., 2008], QUICK [Molemaker et al., 2008] and CIP methods [Kim et al., 2008a] improve the accuracy of the semi-Lagrangian advection to obtain second- or third order accurateness. The PIC/FLIP approach [Zhu and Bridson, 2005] which is popular in industry productions uses an additional particle set for a more accurate advection. Mullen et al. [2009], on the other hand, propose an implicitly energy-preserving velocity integration scheme for tetrahedral meshes.

Alternatively, Fedkiw et al. [2001] advocate detecting and amplifying existing vortices to combat dissipation. This is extended to multilevel confinement by Jang et al. [2010]. Similarly, manually seeded vortex particles can be used to reinforce turbulence vortices and combat dissipation [Selle et al., 2005].

While these methods help to reduce the numerical dampening, the detail that can be represented is still inherently limited by the underlying grid resolution.

2.2.2 Sub-grid Methods

Another way to combat numerical dissipation is to adaptively refine the simulation grid in critical areas. Losasso et al. [2004] use an adaptive octree structure to discretize the simulation grid, while the method by Feldmann [2005] operates on unstructured meshes. The combination of two-dimensional and three-dimensional simulations [Irving et al., 2006], [Chentanez and Mueller, 2011] has also successfully been used to represent large bodies of fluids. The computational overhead introduced by the adaptivity however only pays off if the detailed motion is confined to only a small part of the simulation space, or for very high resolutions.

A different approach to obtain sub-grid accurate results is to track the visible quantity, e.g. smoke or liquid using Lagrangian markers. Traditionally, these fields are represented using Volume-of-Fluid [Hirt and Nichols, 1981] methods in Engineering and density fields or level-sets in Graphics. For liquids, particle level-sets [Enright et al., 2002a] increase the resolution of a level-set using Lagrangian markers. Bargteil et al. [2006] and Wojtan et al.[2010] use a triangle mesh to represent and track liquid-air interfaces. Brochu et al. [2010]

use Voronoi diagrams to generate a surface sub-grid accurate meshes for liquids. A triangle mesh representation to represent the smoke/air surface for plumes has been proposed by Brochu et al. [2009]. Particle representations are another popular choice, but large numbers are usually necessary to represent dense surfaces without noise. While the Lagrangian markers in these methods allow for the detailed representation below grid scale, the dynamics are still limited by the grid resolution of velocity field. Similar in spirit to the approaches presented in this thesis, Thuerey et al. [2010] take advantage of the Lagrangian representation and compute sub-grid surface tension dynamics directly on a air/liquid interface mesh.

A major source of turbulent detail is the interaction of fluids with solids. By improving the accuracy of boundary dynamics or modeling the interaction, higher detail levels can be achieved for the fluid system. Two-way coupling of fluids has been addressed by Carlson et al. [2004], Guendelmann et al. [2003] and Klingner [2006]. More recently researchers have modeled subgrid interactions with objects more accurately through the use of apertures [Batty et al., 2007; Robinson-Mosher et al., 2008]. Nevertheless, very little previous work in graphics addresses the problem that the thin turbulent boundary layer is not resolved in the simulation, resulting in turbulence not being shed.

2.2.3 Real-time Fluid Simulation

There are only few methods that enable detailed fluid simulation at interactive frame rates. Crane [2007] demonstrated the realization of three-dimensional Eulerian fluid solvers on a GPU, while Cohen et al. [2010] use a multigrid GPU based-solver to efficiently solve the Navier-Stokes equations in real-time. However, fine-grained turbulent detail is hard to achieve with these direct approaches. Treuille et al. [2006] presented a method to precompute reduced bases of flows, enabling simulations at real-time frame rates. This approach is extended by Wicke et al. [2009] to couple precomputed fluid tiles, which allows the simulation of large scenes. As the method requires large amounts of memory for complex scenes, it is difficult to apply in interactive scenarios. Horvath et al. [2009] use a hybrid particle approach with a coupled 2D and 3D simulation to efficiently simulate fire simulations on the GPU using a fixed camera perspective. Chentanez et al. also combine two- and three-dimensional techniques [Chentanez and Mueller, 2011] or couple a grid-based simulation with particle systems [Chentanez and Müller, 2010] to enable real-time simulation of open water channels and ocean surfaces. Real-time simulations of particle based liquids have been demonstrated in [Müller et al., 2005], but as these simulations heavily depend on neighborhood

calculations, detailed simulations can be very expensive, and can be difficult to stabilize.

2.3 Lagrangian Vortex Methods

Instead of solving the Navier-Stokes equations in the velocity space, they can also be solved in vorticity space. Vorticity describes flow rotation, and is an equivalent description of fluid motion. The vorticity equation can be evaluated on regular meshes and grid using FDM or FEM formulations with similar accuracy and performance as the Navier-Stokes equation. Boundary conditions, obstacle interaction and free surfaces are however more efficiently treated in a velocity representation. Purely Eulerian vortex methods are therefore rarely used in practice. On the other hand, vortex methods have the desirable property that rotational flow features such as eddies are more compactly represented than in the velocity formulation. This makes Lagrangian or hybrid vortex methods an attractive choice for strongly rotational flows, especially those with strong turbulence generation. In this section, an overview of Lagrangian vortex methods in CFD is given.

Lagrangian Primitives The most general and commonly used primitive for vortex methods is the *Vorton*, a point representation of vorticity. Vorton methods in two dimensions have been discussed as early as 1931 [Rosenhead, 1931]. In three dimensions, vortex dynamics are much more involved as a vortex stretching term appears, which is hard to stabilize in a Lagrangian setting. The first three-dimensional methods appeared therefore much later [Beale and Majda, 1982] with the convergence being proven by Hald [1979]. A stable treatment of the vortex stretching term still remains one of the main challenges of vorton methods. To this end, hybrid methods employing an additional grid representation [Marshall and Grant, 1996] have been successfully used to stabilize this term. Vortex filament methods [Leonard, 1975; Leonard, 1980], which discretize vorticity using one-dimensional space curves, do not share this problem as the vortex stretching term vanishes in their motion equations. However, this comes at the price of ever-increasing geometry due to re-meshing of the connected elements [Chorin, 1981]. Although global re-meshing using vorticity transfer has been proposed to partly reduce geometric complexity [Lindsay and Krasny, 2001] this is still a largely unsolved problem which restricts simulation run-length and complexity. Finally, vortex sheets, a representation of vorticity on two-dimensional surface, have successfully been used to simulate vorticity dynamics at interfaces. While early methods used points or filaments to discretize the vortex sheet surface

[Agishtein and Migdal, 1989], more recent methods employ a mesh of triangular [Brady et al., 1998] or quadrangular surface elements [Lozano et al., 1998; Stock et al., 2008]. Vortex sheets share the advantages and drawbacks of filaments, but their two-dimensional connectivity makes them very suitable for interface transition modeling, while filaments are often employed for problems including vortex ring breakup.

Extensions A common problem of all Lagrangian vortex methods is the handling of diffusion. Two types of solutions have been proposed, firstly core-spreading [Leonard, 1980], in which the primitives kernel is widened, and second vorticity redistribution between neighboring primitives [Gharakhani, 2003]. While these methods have been used successfully for densely sampled vorton simulations, they are largely unsuitable for sparse simulations and the higher dimensional filaments and vortex sheets. Therefore, Lagrangian vortex methods are often applied to problems where diffusion is not relevant and can be omitted, e.g. turbulent flows.

Another big issue is the complexity of the velocity evaluation. Classically, the velocity is obtained by applying the Biot-Savart law between primitives, resulting in a complexity of $\mathcal{O}(n^2)$. Treecodes [Ploumhans et al., 2002; Wang, 2004] employ spatial acceleration structures to increase performance, while fast multi-pole methods [Winckelmans et al., 1996; Dehnen, 2002] use far-field approximations of the Biot-Savart law to compute distant interactions more efficiently. Another approach is the Vortex-in-Cell (VIC) method [Cottet and Koumoutsakos, 1999; Cottet and Poncet, 2003; Stock et al., 2008], which projects vorticity to a grid, and solves a Poisson problem to obtain velocity. This however induces a strong regularization for moderate grid resolutions, which may not be desired.

Finally, extensions to vorticity generation have been proposed. While vorticity generation at obstacles may be represented using boundary conditions, baroclinic generation has to be included as a source term. This theory was proposed by Meng [1978] and studied by Tryggvason et al. [1983].

Sparse Sampling In Computer Graphics, vortex methods are mainly used as a sparse representation. Instead of completely discretizing the motion field, vortex primitives are often used to reinforce eddies, and augment a base simulation. Selle et al. [2005] uses vortons to reinforce bulk turbulence on a Eulerian base simulation. Filament methods, on the other hand, have been used to control a fluid simulation by manually adding vortex rings [Angelidis and Neyret, 2005; Angelidis et al., 2006]. Vortex sheet theory is

used by Kim et al. [Kim et al., 2009] to reinforce the breakup of liquid sheets. However, they discretize the vortex sheet on the grid, and synthesize motion using Eulerian vorticity confinement. While vortex methods are increasingly being used in Graphics, the research base is still very weak, compared to both velocity methods and vortex methods in CFD. Most application in Graphics simply add vortons manually to obtain a more turbulent look and feel.

2.4 Turbulence Methods

The high numerical cost involved in the simulation of small-scale turbulence has given rise to turbulence modeling approaches. These models separate turbulent fluctuations from the mean flow, and aim to describe these fluctuations in a statistical manner, based on properties obtained from the large-scale mean flow. Turbulence modeling methods have a long tradition in Engineering. They are usually employed to estimate the influence of turbulence on the mean flow, e.g. the forces exerted on airplane wings, or the flux reduction in pipe flow. In Computer Graphics, on the other hand, the main interest is not obtaining corrected mean flows, but to obtain the transient turbulent detail itself to enhance visual detail. Therefore, turbulence synthesis methods are employed to generate artificial detail. Advanced synthesis methods use turbulence modeling methods as an estimator to predict the turbulence distribution to synthesize. Below, an overview of turbulence modeling and turbulence synthesis models is given.

2.4.1 Turbulence Modeling

Classical turbulence models in CFD model turbulent viscosity, that is the virtual diffusion turbulence induces on the mean flow [Prandtl, 1945]. The simplest classical turbulence models which are able to describe non-trivial flows are mixing length models [Smagorinsky, 1963], [Baldwin and Lomax, 1978]. Mixing length models have been used successfully to describe boundary layer flow in e.g. the aerospace industry. However, they rely on manual specification of the mixing length, which is scene-dependent and only known for certain types of flows. To alleviate this issue, complete convective-diffuse models have been proposed, which model the dynamic of turbulence using a set of differential equations without the need for a scene-dependent specification. The Spalart-Almaras models [1994], which is used in aeronautic applications, directly models turbulent viscosity using one PDE. Two-equation models, on the other hand, model the evolution of parameters such as turbulent kinetic energy and dissipation using two PDEs. The k–ε model

[Launder and Sharma, 1974] and the k–ω model [Wilcox, 1993] are still the most commonly used turbulence models in Engineering today, and included in most CFD applications due to their generality and simplicity. However, the prediction quality and stability of these models strongly depends on the type of flow. Therefore, extensions have been proposed for e.g. better stability in wall-regions [Jones and Chen, 1994], to cite one example.

The class of classical turbulence models has however two fundamental limits. First, it relies on Reynolds averaging to separate mean flow and turbulence, which might not be sufficient, especially to describe turbulence transition and coherent eddies. To this end, Large Eddy Simulations (LES) have been introduced [Smagorinsky, 1963]. LES models use more accurate frequency filters for separation, and directly simulate coherent eddies. Originally, LES was used to describe internal flows in meteorology, but has been adapted for more general applications in CFD as well [Haworth and Jansen, 2000]. A recent extension to LES are Detached Eddy Simulations (DES) [Spalart, 2009] which are even more suitable to represent coherent eddies. The second limit concerns the information provided by classical models. While turbulent viscosity is sufficient to describe the virtual diffusion of the mean flow, is does not provide information on flow anisotropy and energy exchange. Reynold stress transport models solve this by modeling not only turbulent viscosity, but the complete turbulent stress tensor [Launder et al., 1975], [Chung and Kim, 1995]. Even further, probability density functions such as the General Langevin Equation [Pope, 1983] directly describe higher-order statistical properties of turbulence using a particle method, while Elliptic Relaxation methods [Durbin, 1993] replace the local convection-diffusion processes in turbulence models by global optimization. While all these methods have higher prediction power than classical turbulence models in theory, they also come at the price of higher complexity, numerical cost and tuning effort. Therefore, they are mainly used in the context of CFD research and high fidelity simulation in e.g. meteorology. With the increasing computation power in the recent years, especially LES and Reynolds stress transport models are however slowly becoming more popular for standard applications.

While the fundamentals of CFD turbulence modeling were developed in the 1970s and 80s, this does not imply no progress has been made in recent years. The topics have however shifted from general purpose turbulence modeling towards more specific topics, as it is common in mature fields. Recent topics include the modeling of turbulence in compressive flows [Aupoix, 2004] or turbulence transition modeling [Langtry et al., 2006], [Dandois et al., 2007], to cite a few examples. A very interesting recent trend is the development of multi-scale turbulence modeling techniques using LES [Chaoat and Schiestel, 2007] or scale adaptive simulation (SAS) [Menter and Kuntz, 2005].

2.4.2 Turbulence Synthesis

Early turbulence methods in Computer Graphics used uniform fields of synthetic turbulence to augmented or replaced basic fluid simulation with synthetic turbulence. Stam [1993] introduced a method that used a Kolmogorov spectrum to produce procedural divergence free turbulence. This approach was used to model nuclear explosions and flames [Rasmussen et al., 2003; Lamorlette and Foster, 2002]. Bridson et al. [2007] suggest taking the curl of vector noise fields to produce divergence free velocity fields. They explicitly address computing flows around objects efficiently by modulating the potential field. These methods however do not take into account the spatial and temporal distribution and dynamics of turbulence, and may lead to unrealistic results for complex flows.

Recent methods use a more complex estimation of flow statistics to better capture the characteristics of turbulence. Kim et al. [2008b] use wavelet decomposition to determine local turbulence intensities, and synthesize turbulence using frequency-matched curl noise. This approach assumes that the base solver can resolve the turbulence dynamics, which is not always the case, e.g., for complex scenes or very coarse solver resolutions. Schechter [2008] and Narain [2008] use transport models to derive turbulence parameters. While this improves the turbulence dynamics, they need to significantly simplify the energy transport, as in Schechter [2008], or make strong assumptions, such as mixing-length models to close their one-equation energy model [Narain et al., 2008]. The combination of complete turbulence models and synthesis is introduced by our work in § 4.

2.5 Recent works

Turbulence methods and vorticity representations for fluid simulations are an active area of research in Computer Graphics. It is therefore not surprising that a number of works on these topics has been published concurrent to or as a follow-up work to the methods presented in this thesis. These approaches will be discussed below.

Turbulence Synthesis While this thesis focusses on synthesis based on curl-noise textures and vorticity, alternative methods for synthesis have been presented in recent years. Chen et al. [2011] present a hybrid turbulence method, which synthesizes turbulence on a particle system, based on our work in § 4. Instead of using a RANS-based turbulence model, however,

they follow a pdf approach and solve the Langevin equation [Pope, 2000] on the particle system. The resulting velocity update is directly used as detail motion of the particles. This avoids both the coherence problem of curl-noise synthesis and the stability issues of two-equation turbulence models employed in our model. However, by interpreting the stochastical Langevin velocities as actual particle motion, the synthesized motion will neither be divergence-free nor spatially coherent. We found these two properties to be vital for producing the characteristic swirly look of turbulent fluid motion and therefore a realistic appearance.

Zhao et al. [2010], on the other hand, propose the use of random forcing on an up-sampled grid as a synthesis method. The divergence-free force fields are precalculated and follow a given energy spectrum. This representation does not require a separate set of markers such as vortex primitives or texture coordinates for curl-noise synthesis. However, the synthesized forces can not directly be animated; instead, alternating pre-computed fields are applied. This temporal inconsistency will lead to artifacts and flow disturbance if the method is used beyond small scales. Still, this method might provide a very attractive alternative for synthesis, if extended such that force fields can be updated at runtime.

Vorticity methods The original vortex particles paper [Selle et al., 2005] was extended by Yoon et al. [2009] concurrently to our work in § 5. Very similar to our approach, they applied the vortex particles onto an up-sampled higher resolution grid. Particles were however seeded manually, as the method did not include a turbulence predictor, which was the focus of our method. Kim et al. [2012] extended the vortex particle approach for buoyant sources. The source terms are calculated on the grid and then mapped to the vortex particle system.

Weissmann et al. [2010] proposed a simulation driven entirely by vortex filaments, with source terms for obstacle-induced shedding. They are able to represent flows with a medium level of turbulence in a very compact fashion, which makes the method very efficient in this regime. This approach was recently extended by an improved re-sampling scheme for filaments, which limits the geometry growth [Barnat and Pollard, 2012]. An introduction to the vortex primitives of particles, filaments and sheets and a comparison of their strengths and weaknesses can be found in § 3.1.2 and § 7.2.

Finally, Brochu et al. [2012] presented a vortex sheet method which is very similar to our work in § 6. The main difference is the replacement of our local evaluation scheme with a fast multipole method (FMM) to accelerate the

Biot-Savart velocity integration. This has the advantage that no Eulerian base simulation is necessary, which obviates the need to tune the grid's buoyancy terms to the mesh-based buoyancy and avoids artifacts induced by the scale separation. On the other hand, it is hard to represent inflows, source terms, obstacle interaction and the coupling to e.g. turbulence methods in this pure vorticity formulation, while these effects are trivial to achieve using our hybrid model. It therefore depends on the application scenario which of the methods is best suited.

CHAPTER 3

Theory and Numerical Methods

This chapter discusses elements of fluid mechanics theory and computational methods for its numerical treatment. It will serve as the theoretical basis for the later chapters, and focusses on topics relevant to our novel methods in the later part of the thesis. The chapter is divided into two blocks, fluid simulation and turbulence theory. We first introduce methods for fluid simulation in its velocity and vorticity form (§ 3.1). Then, turbulence theory is summarized (§ 3.2) and approaches for modeling turbulence properties are introduced (§ 3.3). Finally, we discuss the turbulent energy spectrum (§ 3.4) and methods to synthesize turbulence based on statistical properties (§ 3.5).

3.1 Fluid dynamics

Fluid dynamics describe the behavior of gases and liquids by the means of continuum mechanics. The equations of motion are derived by the application of Newton's second law, in accordance to solid mechanics. In constrast to solids or deformables however, fluids do not sustain shearing stresses without continuously deforming, and do not have a rest state. Therefore, the modeling equations and numerical methods employed differ from solid mechanics, although it is technically possible to treat fluids using stress-strain relations as used in the modeling of deformable bodies.

Velocity formulation The motion equations for Newtonian fluids in its continuum formulation are called the Navier-Stokes (NS) equations. They are a set of partial differential equations in the velocity field $\mathbf{u}(\mathbf{x},t)$. For the case of incompressible fluids, they are written as

$$\nabla \cdot \mathbf{u} = 0 \tag{3.1}$$

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \frac{1}{\rho}\mathbf{g} \quad , \tag{3.2}$$

with the fluid viscosity ν, density ρ, and the gravity $\mathbf{g}$. The pressure field is denoted by p. Eq. (3.1) is called *continuity equation*, and ensures conservation of mass. The *momentum equation* Eq. (3.2) consists of several parts. The first term on the right-hand side of the equation is the pressure correction. The pressure field p effectively counteracts fluid compression and gravity, and is chosen such that the continuity equation is fulfilled. The remaining terms describe the diffusion introduced by viscosity and gravitational and external forces.

The left-hand side of the momentum equation is the substantial derivative of $\mathbf{u}$, defined as

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{d}t} = \frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} \quad . \tag{3.3}$$

From an Eulerian viewpoint, this implies a self-advection of the velocity field, while in a Lagrangian representation this term is implicitly handled by the movement of the primitives.

Vorticity formulation Vorticity is defined as the curl of the velocity field $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ and is a description of flow rotation. This representation is advantageous especially for describing turbulent flows, which consist of small rotational whirls, as these structures have a more compact support in vorticity than in velocity formulation. By applying the curl operator to the NS equations, we obtain the *vorticity equation*

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{d}t} = \boldsymbol{\omega}\cdot\nabla\mathbf{u} + \nu\nabla^2\boldsymbol{\omega} + \frac{1}{\rho}\nabla\rho \times \left(\mathbf{g} + \frac{1}{\rho}\nabla p\right). \tag{3.4}$$

The substantial derivative on the left-hand side includes vorticity advection, while the right-hand side consists of the vortex stretching, diffusion and baroclinity terms. Vortex stretching describes the deformation of the vorticity field under the influence of the velocity field, while baroclinity models its behavior across density gradients and gravity. One implication of this baroclinity term is buoyant movement, which in the vorticity formulation is represented by a

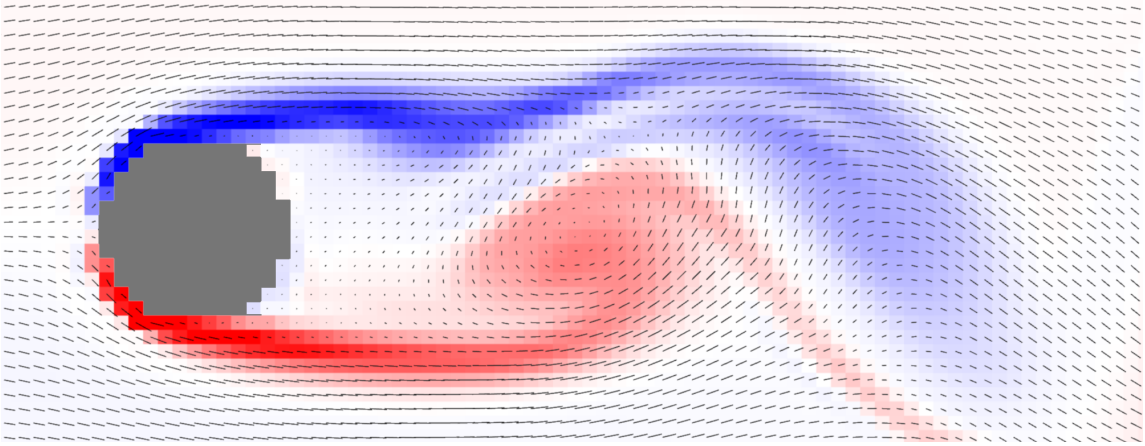


Figure 3.1: *A simulation of the Karman Vortex Street, the flow around a cylinder, is shown. Velocity vectors are drawn in black, vorticity in red (positive) and blue (negative). Concentrated vorticity, as in the middle of the image, describes a vortex. The elongated streaks of vorticity, so-called vortex sheets, separate flow regimes and indicate the boundary layer around the obstacle.*

sheet of vorticity at the density gradient or interface, yielding e.g. the typical vortex rings for rising smoke.

The continuity equation Eq. (3.1) on the other hand is implicitly satisfied, as rotational fields are divergence-free. We also note that the pressure term vanishes for flows with constant densities. However, advection and vortex stretching require a velocity field, which has to be obtained by integrating the vorticity field.

As the Helmholtz theorem shows, we can decompose a given velocity field $\mathbf{u}$ into a curl-free component $\mathbf{u}_\Phi$ and a divergence-free component $\mathbf{u}_\Psi$. The divergence-free component can be related to a vector potential $\mathbf{\Psi}$ and the vorticity by

$$\mathbf{u}_\Psi = \nabla \times \mathbf{\Psi} \,, \qquad \boldsymbol{\omega} = \nabla \times \mathbf{u}_\Psi \quad .$$

Applying the continuity equation for incompressible fluids, we find that $\mathbf{u}_\Phi$ has to be constant. Therefore the velocity field is completely described by vorticity or the vector potential $\mathbf{\Psi}$. This is strictly only true for the free-space case, however. With limited domains, as encountered in all grid-based simulations, additional terms are needed to describe the behavior at the domain boundaries, if velocity does not vanish there. Finally, to integrate the velocity field induced by vorticity we can use the free-space solution to the rotation operator, the Biot-Savart law

$$\mathbf{u}_\Psi(\mathbf{x}) = \frac{1}{4\pi} \int \boldsymbol{\omega}(\mathbf{x}') \times \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} \, d\mathbf{x}' \quad . \tag{3.5}$$

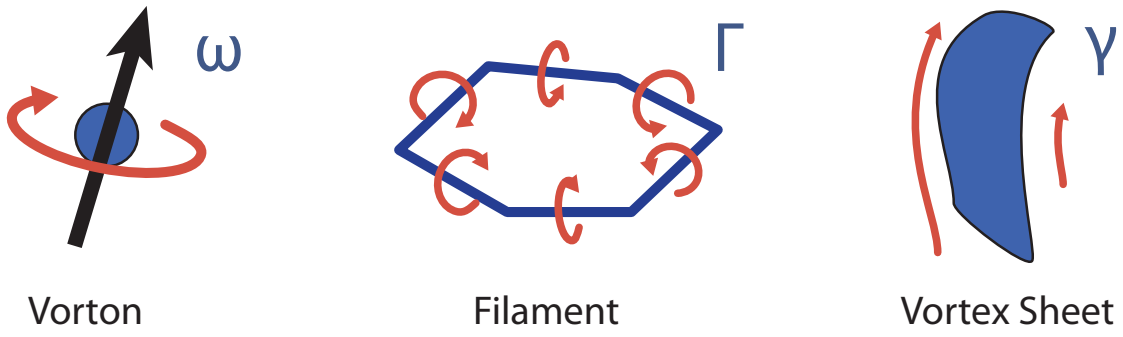


Figure 3.2: *Three different Lagrangian primitives to represent vorticity are shown, with their induced velocity marked in red. Vortons are particles which induce a rotation around the axis of their associated vorticity $\boldsymbol{\omega}$. One-dimensional curve primitives are called filaments. They induce circular motion around the curve tangent based on a circulation number Γ. Vortex sheets contain the vorticity γ confined to a surface. They represent a velocity jump between two flow regimes.*

3.1.1 Eulerian discretization

There are several ways to solve the NS equations. In Computer Graphics, the most common approach are Eulerian solvers with semi-Lagrangian advection as introduced by Stam [1999]. The velocity field is discretized on a Marker-and-Cell (MAC) grid, and operator splitting is applied to Eq. (3.2) to handle the individual terms separately. The semi-Lagrangian advection scheme is unconditionally stable, but introduces artificial dampening, the so-called *numerical viscosity*, which is much higher than the natural fluid viscosity. Therefore, the viscosity term in Eq. (3.2) is neglected. To obtain the pressure field, a Poisson equation is solved which is the most costly step of the simulation. A detailed account on implementing Eulerian fluid solvers can be found in the book by Bridson [2008].

In the same manner it is possible to solve the vorticity equation on a regular grid. In addition to velocity, vorticity is sampled on a grid, and integrated over time using the vorticity equation Eq. (3.4). Additionally, the vector Poisson equation $\boldsymbol{\omega} = \nabla^2 \boldsymbol{\Psi}$ is solved in each step to obtain the vector potential, and by derivation the velocity field $\mathbf{u}_{\boldsymbol{\Psi}}$.

3.1.2 Lagrangian primitives

There are also pure Lagrangrian approaches to solve the NS equation in its velocity form. The most commonly used method in Computer Graphics is *Smoothed Particle Hydrodynamics* (SPH), which is especially suitable for free-surface problems. SPH uses particles with a smooth density kernel to represent fluid properties and solves pressure correction using particle-particle interaction. This method is however not the focus of this thesis and the reader is referred to [Monaghan, 2005] for an overview of SPH.

Lagrangian solutions for the vorticity equation are common in CFD, and becoming increasingly popular in Computer Graphics. This thesis is heavily based on Lagrangrian methods for vorticity, therefore the theory is expanded here in more detail.

There are two ways in which Lagrangian vortex elements can be used. On the one hand, they can be used to discretize the complete vorticity field as an alternative representation to velocity. On the other hand, sparse elements can be used to augment an existing simulation. The second approach can increase performance by focusing resolution in desired areas. Also, remeshing inaccuracies are not as critical since the underlying simulation provides consistency. However, the interplay between the vortex primitives and the underlying simulation is nontrivial. In particular, vorticity structures need to be mutually exclusive in both simulations, to avoid injecting excess energy into the system.

Below, we will introduce three types of Lagrangian primitives for discretizing vorticity. *Vortons* are zero-dimensional point elements, *Filaments* represent vorticity confined to a one-dimensional curve, while *Vortex sheets* describe vorticity on a thin two-dimensional surface. Fig. 3.2 visualizes these primitives. In theory, a given vorticity field can be discretized equally well by all three primitives. However, each of them has inherent advantages and disadvantages as far as re-meshing, motion equation and connectivity are concerned. Also, some forms allow a more natural representation of a certain flow geometry than others. It is e.g. possible to discretize vorticity on a thin surface using a patch of filaments. However, it is much easier to ensure a uniform coverage of a surface deforming in the flow by representing it using vortex sheets. A detailed comparison of vortex primitives can be found in [Stock, 2006] and [Cottet and Koumoutsakos, 1999].

Vortons

The most popular and well-researched primitive is the point element *Vorton* [Hald, 1979]. Each vorton i owns a position $\mathbf{x}_i$ and its associated vorticity $\boldsymbol{\omega}_i$. The total vorticity field of the system is given by

$$\boldsymbol{\omega}(x) = \sum_i \boldsymbol{\omega}_i \delta(\mathbf{x} - \mathbf{x}_i) \tag{3.6}$$

using Dirac's delta function δ. In many methods, an additional particle radius or kernel is used. Vortons are the most general primitive and are especially suitable for highly turbulent flows. In these flows, coherent structures break down to small isolated vortices, and the role of connectivity is diminished.

Dynamics The motion equation for the vortons is given by Eq. (3.4). While advection is handled implicitly, the right-hand side terms are not easily expressed in terms of a particle system. The vortex stretching term can be evaluated by explicitly calculating the velocity gradient tensor at the particle position. However, this has been shown to produce divergent flow fields [Cottet and Koumoutsakos, 1999]. One way to regularize this problem is to use an intermediary grid [Marshall and Grant, 1996]. The diffusion term can be implemented by core-spreading, that is increasing the particles radius [Leonard, 1980] or particle strength exchange methods [Degond and Mas-Gallic, 1989]. Full vorton models require overlapping particles for convergence, therefore re-meshing is needed to ensure full coverage. While local re-meshing methods do exist [Shankar and van Dommelen, 1996], most vorton methods use a form of global re-meshing [Cottet and Koumoutsakos, 1999].

Velocity integration The most direct approach to obtain a velocity field from a vorton system is the discretization of Eq. (3.5)

$$\mathbf{u}_{\boldsymbol{\Psi}}(\mathbf{x}) = \frac{1}{4\pi} \sum_i \boldsymbol{\omega}_i \times \frac{\mathbf{x}_i - \mathbf{x}}{|\mathbf{x}_i - \mathbf{x}|^3} \quad . \tag{3.7}$$

This equation is however singular for points close to vortons. Chorin et al. [1973] therefore introduced a regularization mechanism

$$\mathbf{u}_{reg}(\mathbf{x}) = \frac{1}{4\pi} \sum_i \boldsymbol{\omega}_i \times \frac{\mathbf{x}_i - \mathbf{x}}{(|\mathbf{x}_i - \mathbf{x}|^2 + \alpha_R^2)^{\frac{3}{2}}} \quad . \tag{3.8}$$

The regularization parameter α_R effectively controls the minimal size of the generated vortices. As an alternative to direct integration, the particles can

be projected on an auxiliary grid using a smoothing kernel. On the grid, the velocity can be obtained by solving the Poisson equation for the vector potential as in the Eulerian case. This hybrid method is called *Vortex-in-Cell* (VIC) and can be used for all vortex primitives [Cottet and Poncet, 2003]. The grid and particle projection kernel act as an implicit regularization, so no additional terms are needed.

Sparse Vortons In Graphics, Vortons are called *Vortex Particles*, and mostly used in a sparse setting. Therefore, less strict re-meshing is used. Also, the diffusion term is commonly ignored, for the same reasons as in the velocity form of the NS equations. The original approach for vortex particles [Selle et al., 2005] does not use velocity integration using the Biot-Savart law, but a confinement force that acts on the underlying simulation, in a similar manner as vorticity confinement [Fedkiw et al., 2001]

$$\mathbf{F}(\mathbf{x}) = \epsilon \sum_i \frac{\mathbf{x}_i - \mathbf{x}}{|\mathbf{x}_i - \mathbf{x}|} \times \boldsymbol{\omega}_i \delta(\mathbf{x}_i - \mathbf{x}) \tag{3.9}$$

where ϵ is the confinement strength. This treatment ensures only vorticity not already present in the underlying simulation is added, but suffers from instabilities if ϵ is chosen inappropriately.

Filaments

Vortex filaments discretize vorticity using one-dimensional line segments or spline curves. Instead of directly storing the contained vorticity on the line elements, an equivalent representation is used. This has advantages for the formulation of the motion equation, as will be shown is the next paragraph. Each segment has an associated circulation number Γ which defines a rotation around the curve segment. It relates to vorticity by

$$\boldsymbol{\omega}(\mathbf{x}) = \Gamma(s)\, \mathbf{t}\, \delta(\mathbf{r}_\perp(s)) \tag{3.10}$$

where $\mathbf{t}$ denotes the line tangent and $\mathbf{r}_\perp$ is the distance perpendicular to the tangent of the curve. Filaments are suitable for e.g. rising smoke with low levels of turbulence, as the characteristic vortex rings arising in such settings are represented naturally using closed filaments.

Dynamics By expressing the vorticity equation in the circulation formulation, we obtain

$$\frac{\mathrm{D}\Gamma}{\mathrm{d}t} = \nu \int_L \nabla^2 \mathbf{u} \cdot \mathrm{d}\mathbf{x} + \frac{1}{\rho} \int_L \nabla p \mathrm{d}\mathbf{x} \tag{3.11}$$

with the line integral of diffusion and baroclinity, respectively. Diffusion is hard to express efficiently for filaments, therefore filaments are mainly used in low-viscous flows, where diffusion can be neglected. This means that for flows without baroclinic generation, circulation effectively remains constant,

$$\frac{\mathrm{D}\Gamma}{\mathrm{d}t} = 0 \tag{3.12}$$

as vortex stretching is implicitly handled by elongation of the primitive itself. This is a very desirable property, as it avoids the problems associated with evaluation of the velocity gradient for vortex stretching in e.g. vorton methods. On the other hand, elongation also leads to ill-shaped line segments, therefore remeshing is necessary. A simple remeshing by subdividing elements that are too long or are strongly curved is often sufficient [Chorin, 1981]. For turbulent flows, however, this means that geometry will increase over time, as complex structures tend to generate even more complex structures on neighboring filaments. For fully-developed turbulence for example, filament strands will inevitably overlay and form a complex intertwined structure which could be represented using a much smaller number of vortons. Some effects of this can be mitigated by hairpin removal techniques [Chorin, 1996] or vortex loop optimization [Weissmann and Pinkall, 2010].

Velocity integration Filaments are integrated by directly evaluating the Biot-Savart law, which for circulation takes the form of the line integral

$$\mathbf{u}_\Psi(\mathbf{x}) = \frac{1}{4\pi}\int_L \Gamma(s)\mathbf{t} \times \frac{\mathbf{x}-\mathbf{r}(s)}{|\mathbf{x}-\mathbf{r}(s)|^3}\mathrm{d}s \quad . \tag{3.13}$$

In this form, $\mathbf{r}(s)$ is the line parameterization and $\mathbf{t}$ is the tangent. Regularization is introduced in the same manner as in Eq. (3.8). Depending on the concrete form of line discretization, flat line segments or spline curves, the discretized form of Eq. (3.13) varies.

Vortex sheets

The description of vorticity confined to a two-dimensional surface is a called a vortex sheet. Vortex sheets are particularly useful to describe thin sheets of vorticity that are induced at flow boundaries in the *boundary layer* and across steep density gradients, between e.g. cold and hot air of a rising plume. This surface is described by a surface mesh, triangle meshes being the most common approach. Vorticity on a surface element is represented using the *vortex sheet strength* $\boldsymbol{\gamma}$, which is defined as

$$\boldsymbol{\omega}(\mathbf{x}) = \boldsymbol{\gamma}(\mathbf{x})\delta(\mathbf{x}) \quad . \tag{3.14}$$

For deriving relations for the vortex strength, it is useful to consider the velocity jump $\Delta\mathbf{u}$ that is induced by the vortex sheet. It relates to vortex strength by $\boldsymbol{\gamma} = \mathbf{n} \times \Delta\mathbf{u}$, with the surface normal $\mathbf{n}$.

Dynamics Using the velocity jump definition, the inviscid transport equation for vortex sheet strength can be derived as

$$\frac{\mathrm{D}\boldsymbol{\gamma}}{\mathrm{d}t} = \boldsymbol{\gamma} \cdot \nabla\mathbf{u} - \boldsymbol{\gamma}(\mathbf{P} \cdot \nabla \cdot \mathbf{u}) - 2\beta_A \, \mathbf{n} \times \mathbf{g} \quad . \tag{3.15}$$

The first term on the right-hand side is the familiar vortex stretching, while the second term describes changes in vortex strength due to elongation in the direction of $\boldsymbol{\gamma}$. Here, $\mathbf{P} = \mathbf{I} - \mathbf{nn}$ is the tangential projection operator. The baroclinity term is expressed using the Boussinesq approximation [Meng, 1978] which is proportional to the Atwood ratio β_A. The Atwood ratio relates the densities of the two fluids to each other, and is defined as

$$\beta_A = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2} \quad . \tag{3.16}$$

The Boussinesq approximation assumes a small Atwood ratio, and is valid for e.g. hot/cold air, but not air/water interfaces. As for filaments, diffusion is not easily modeled for vortex sheets, so they are mostly used in the inviscid limit. Re-meshing is essential for vortex sheet methods as the induced vorticity quickly deforms the sheets. As in the case of filaments, however, the splitting of ill-shaped elements causes a steady increase in geometry for turbulent flows.

Velocity integration The Biot-Savart law for the vortex sheets has the form

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_S \boldsymbol{\gamma}(\mathbf{x}') \times \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} d\mathbf{x}' \quad . \tag{3.17}$$

Again, regularization can be performed as in the case of Vortons. Alternatively, VIC can be used for integration and regularization.

Conversion between representations

The primitives introduced above use different representations for vorticity, namely vorticity $\boldsymbol{\omega}$, circulation Γ and vortex sheet strength $\boldsymbol{\gamma}$. As each representation has different properties as far as dynamics or source terms are concerned, it is sometimes advantageous to convert between representations,

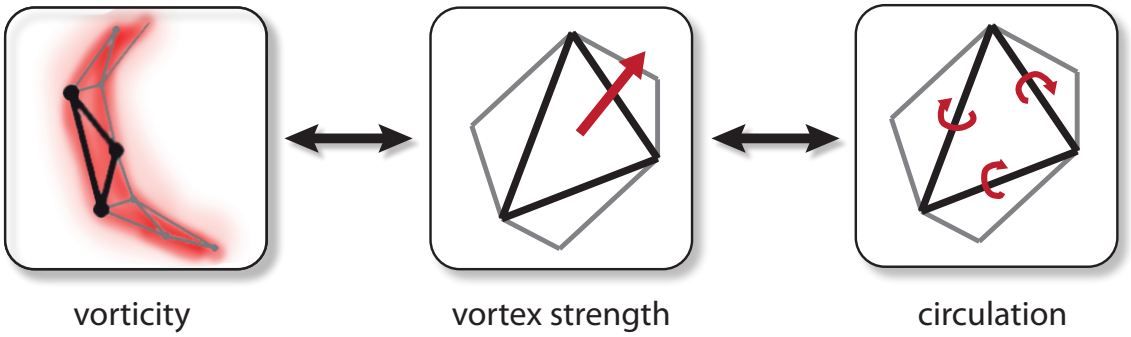


Figure 3.3: *The continuous vorticity field around a surface can be represented in terms of a vortex sheet strength or circulation. Both are stored per surface triangle, and are equivalent representations. Vortex strength is a vector value, while circulation consists of three scalar rotation values around the edges of the triangle.*

and use the one most appropriate for the task. We will focus on vorticity confined to thin sheets here, as this theory is used in § 6.

While vortex sheet strength, discretized using a triangle mesh, is the most natural representation for this case, its dynamic equations are more involved than those of the circulation formulation. According to Stock et al. [2008], the vortex sheet strength vector $\boldsymbol{\gamma}$ of a triangle uniquely relates to the three circulations numbers Γ_i around the triangles edge vectors $\mathbf{e}_i$. We can therefore express the motion equations in terms of both circulation and vortex sheet strength. This is illustrated in Fig. 3.3. It should be noted that the circulation numbers are defined per triangle, which means that adjacent triangles may have different circulation numbers for the same edge. In order to convert to vortex sheet strength, we can use the relation

$$\boldsymbol{\gamma} = \frac{1}{A} \sum_{i=1}^{3} \Gamma_i \, \mathbf{e}_i \tag{3.18}$$

with A denoting triangle area. On the other hand, conversion from vortex sheet strength to circulation can be performed by solving the overdetermined linear system

$$\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \end{pmatrix} = A \begin{pmatrix} \boldsymbol{\gamma} \\ 0 \end{pmatrix} \quad . \tag{3.19}$$

During this conversion process, the vorticity component normal to the surface is lost. For vortex sheets, this is however a desired property [Stock et al., 2008].

3.2 Turbulence

Many complex flows, ranging from chimney smoke and explosions to the wake of a ship in the ocean, show chaotic and irregular vortical flow disturbances. These flows are called *turbulent flows*. Compared to *laminar* flows, such as slow-moving rivers, or the air flow field over a candle, turbulent flows show an abundance of detail on many length scales. While this detail constitutes the appealing look of many flow phenomena, representing it directly in the simulation requires enormous storage and computing capabilities. It is however possible to describe turbulence in terms of their statistical properties. Turbulence modeling theory aims exactly at that. In this section, established turbulence theory is introduced which forms the basis for our turbulence-aware simulations in § 4, § 5 and § 6. A more detailed overview of turbulence theory can be found in the books of Pope [2000] and Wilcox [1993].

3.2.1 The Reynolds Average

To provide a measure of the overall strength of turbulence in the flow, the *Reynolds number*

$$\mathrm{Re} = \frac{vL}{\nu} \tag{3.20}$$

is used. Flows with a Reynold number below 1500 are typically laminar, while a Reynolds numbers over 5000 are a strong indicator for a turbulent flow [Pope, 2000]. The quantities used here are the flow velocity v, the fluid viscosity ν and the characteristic lengthscale L. The definition of L depends on the problem at hand, for pipe flow it would e.g. correspond to the pipe diameter.

While the Reynolds number provides a general estimate on turbulence behavior, most flows are not completely turbulent or laminar. A river with an immersed obstacle, for example, may be laminar in most regions, but show transition to turbulence in the wake of the obstacle. To study the behavior of turbulence, it is therefore beneficial to decompose the flow field in a turbulent and mean flow component. The *Reynolds decomposition* achieves this by introducing a mean velocity field by the average $\bar{\mathbf{U}} = \langle \mathbf{u} \rangle$. The remaining component $\mathbf{u}' = \mathbf{u} - \langle \mathbf{u} \rangle$ then describes the turbulent fluctuation. It can be shown that both components remain divergence-free [Pope, 2000].

The averaging operator $\langle \cdot \rangle$ introduced above is interpreted in the sense of an expectation value of a random field. Its concrete realization, and therefore the concrete classification of turbulence and mean component depends on

the type of problem investigated. For most applications, an average over ensembles, time or lengthscale is used. The equivalence of these interpretations for large sample numbers is given by the ergodicity theorem.

RANS Using the mean flow definition, it is possible to derive properties for the mean and the fluctuating component separately. If we apply the ensemble average operator to the Navier-Stokes equation, we obtain the motion equation for the mean component, the *Reynolds-averaged Navier-Stokes equation (RANS)*

$$\frac{\mathrm{D}\bar{\mathbf{U}}}{\mathrm{d}t} = \nu\nabla^2\bar{\mathbf{U}} - \rho(\nabla\cdot\boldsymbol{\tau}) - \frac{1}{\rho}\nabla\langle p\rangle \quad . \tag{3.21}$$

This equation is identical to the Navier-Stokes momentum equation, except for the additional stress term $\rho(\nabla\cdot\boldsymbol{\tau})$. The symmetric tensor $\tau_{ij} = \langle u_i' u_j'\rangle$ is called the *Reynolds stress tensor*, and describes the influence of turbulent fluctuations on the mean flow field.

The RANS equation is used in many engineering applications, as it allows to predict the impact of small-scale turbulent detail on e.g. the mean flux in an pipe or engine, without directly simulating it. Unfortunately, the Reynolds stress tensor still depends on the fluctuating components, and cannot be expressed in terms of averaged properties. This *closure problem* in CFD is solved by additional assumptions and models of the behavior of turbulence, and has given rise to an area of research, namely *turbulence modeling*.

Turbulent Viscosity Hypothesis To investigate the effect of the Reynolds stress tensor in Eq. (3.21), it is helpful to split the Reynolds tensor into a isotropic and and anisotropic part. The isotropic component can be expressed in terms of the *turbulent kinetic energy*, that is the energy contained in turbulent fluctuations. It is defined as

$$k = \frac{1}{2}\langle\mathbf{u}'\cdot\mathbf{u}'\rangle \quad . \tag{3.22}$$

The isotropic component now becomes a diagonal tensor $\frac{2}{3}k\delta_{ij}$ and can therefore be expressed as a scalar. This means its effect in Eq. (3.21) is that of a pressure, and it can easily be absorbed in an effective pressure $\langle p\rangle_E = \langle p\rangle + \frac{2}{3}k$. The anisotropic component, on the other hand, is defined as $a_{ij} = \tau_{ij} - \frac{2}{3}k\delta_{ij}$ and still needs to be modeled. The *turbulent viscosity hypothesis* assumes that its effects is purely viscous. This is a reasonable approximation, as the superposed small-scale movements act as a diffusion on larger scales. The

turbulent viscosity hypothesis is therefore expressed in analogy to viscous stress by

$$a_{ij} = 2\rho\nu_T S_{ij} \tag{3.23}$$

with the scalar turbulent viscosity ν_T and the mean strain tensor $S_{ij} = \frac{1}{2}(\frac{\partial \bar{U}_i}{\partial x_j} + \frac{\partial \bar{U}_j}{\partial x_i})$. If we substitue the Reynolds tensor and the effective pressure and turbulent viscosity in Eq. (3.21), we obtain

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla\langle p\rangle_E + \nabla((\nu + \nu_T)\nabla\mathbf{u}) \tag{3.24}$$

which has the form of the NS equation Eq. (3.2) with modified pressure and the increased viscosity term $\nu + \nu_T$. We have to note, though, that ν_T is in general a function of space and time, while the molecular viscosity ν is a constant.

If the turbulent viscosity hypothesis is used, the problem of modeling the Reynolds stress tensor reduces to modeling the scalar turbulent viscosity ν_T. Most classic turbulence models are based on this assumption. However, it has to be noted that turbulent viscosity is only an approximation, and cannot describe certain effects, such as anisotropic turbulence generation. Turbulence models based on turbulent viscosity are discussed in the next chapter. A short outlook on including anisotropic effects will also be provided.

3.3 Turbulence Modeling

Turbulence modeling tries to predict properties of the fluctuating turbulence based on the mean velocity field. For averaged simulations such as RANS, the important variable to model is the Reynolds stress tensor, and therefore a big part of existing research in turbulence focuses on predicting the Reynolds stress as accurate and general as possible. In this section, classical turbulence modeling based on turbulent viscosity is introduced. This is what is used in most CFD simulation packages for engineering, and it also forms a basis for our methods in § 4, § 5. There are also other, fundamentally different approaches for describing turbulence, most notably stochastic pdf methods [Pope, 1983] and Large Eddy Simulations (LES) which are used heavily in e.g. meteorology. An overview of LES methods can be found in [Galperin and Orszag, 1993] and [John, 2006].

3.3.1 Energy Transport Models

There are different approaches for modeling turbulent viscosity, most of which are based on empirical assumptions. These methods can be classified by the number of model variables and in terms of their completeness, that is whether they require scene-dependent constants or fields. The simplest conceivable model is assuming ν_T to be constant across the flow. This model is limited to very simple flows, and does not provide much insight over directly specifying turbulence energy. It is thus not useful for any practical application.

A better approach is to model ν_T in terms of a mixing length. These models can be accurate if the mixing length of the respective problem is known, and have successfully been used in aerospace engineering [Baldwin and Lomax, 1978]. Other models use the fact that turbulence properties are well described by advection-diffusion processes, and model these processes using one or more partial differential equations. Due to their generality, these models are among the most common turbulence models. Below, a mixing-length model and one common two-equation model are presented.

Mixing length model The *mixing length model* is based on a generalization of the explicit turbulent viscosity term from boundary layer flows. Baldwin et al. [1978] suggest the definition

$$\nu_T = l_m^2 \|\mathbf{\Omega}\| \tag{3.25}$$

where l_m is the characteristic mixing lengthscale, and $\Omega_{ij} = \frac{1}{2}(\frac{\partial \bar{U}_i}{\partial x_j} - \frac{\partial \bar{U}_j}{\partial x_i})$ the rotation tensor of the mean flow field. The mixing length encodes the geometry of the problem, and the accuracy of the model largely depends on the correct specification of this length. Analytic expressions for l_m are known for a certain type of problems, such as its linearity in wall distance in the log-law region of boundary layers. On the other hand, for the general case far from boundary layers, the mixing length behavior is largely unknown. Therefore, this model is considered *incomplete*.

k–ε model The k–ε model uses a similar definition of turbulent viscosity as mixing length models, but expresses it in terms of turbulent kinetic energy k and turbulence dissipation ε as

$$\nu_T = C_\mu \frac{k^2}{\epsilon} \quad . \tag{3.26}$$

The modeling constant C_μ is defined as 0.09 from empirical observation [Launder and Sharma, 1974]. An evolution equation for the variables k, ε could theoretically be obtained by a Reynolds-average of the Navier-Stokes equation in the same way as Eq. (3.21). As this equation contains mainly terms that cannot be derived from mean flow properties, however, the implication of the individual terms is studied and modeled. The complete PDE system in this model is defined as

$$\begin{aligned} \frac{\mathrm{D}k}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_k}\nabla k) + \mathcal{P} - \varepsilon \\ \frac{\mathrm{D}\varepsilon}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_\varepsilon}\nabla \varepsilon) + \frac{\varepsilon}{k}(C_1\mathcal{P} - C_2\varepsilon) \quad . \end{aligned} \tag{3.27}$$

The terms $\mathcal{P}$ and ε denote production and dissipation of turbulent kinetic energy respectively, while the modeling constants are specified as $\sigma_k = 1$, $\sigma_\varepsilon = 1.3$, $C_1 = 1.44$ and $C_2 = 1.92$. It can be observed that both equations consist of a turbulent diffusion term in analogy to the RANS equation Eq. (3.21) as well as advection and production and dissipation terms. The implicit advection contained in the substantial derivative on the left-hand side refers to advection in the mean flow field $\bar{\mathbf{U}}$. The production term depends on the strain of the mean flow field and is deduced as

$$\mathcal{P} = 2\nu_T \|\mathbf{S}\|^2 \quad . \tag{3.28}$$

The equation system is now fully specified and only depends on properties of the averaged flow field, and is therefore considered *complete*. It is due to this generality that the k–ε model is among the most commonly used turbulence models in CFD.

3.3.2 Extending Energy Transport Models

Turbulence models are, due to their semi-empirical nature, only accurate under certain conditions. A multitude of turbulence models exist, and is used depending on the scenario. For example, the k–ε model performs well for shear flows with small pressure gradient – for strong pressure gradients, the k-ω model is superior, but has other drawbacks. In general, RANS turbulence modeling is considered much less accurate than using Reynolds stress transport models, pdf methods or LES. On the other hand, it is by far the best understood approach, easy to implement and most importantly, computationally inexpensive. For the detail level desired in typical Graphics applications, LES for example does not gain much over direct simulation in terms of performance. Therefore, RANS is still the most commonly employed method, even in CFD where accuracy is of more importance than in Graphics.

Stability To address some of the shortcomings of RANS-based turbulence methods, model extensions have been proposed. For our purposes, the most vital issues is to address to instability of the k–ε model for low values of k and ϵ. This model therefore requires a minimal turbulence intensity to be present. But even a simulation with high turbulence level may become instable in wall regions, as the viscous sublayer very close to the wall drives the effective Reynolds number to zero. The k–ε model is therefore often extended by *Low-Reynolds models*, which consist of additional dampening terms that are active in near-wall regions. For e.g. real-time simulations with large timesteps, however, even this may however not be sufficient as the simulation can still easily become instable. An alternative is a clamping system that restricts the parameters to a meaningful range. Such a system is described in § 4.3.

Reynolds Stress Transport Models A further limitation of all RANS-based turbulence models is their reliance of the turbulent viscosity hypothesis. The assumption implies that the Reynolds tensor is aligned to the mean flow strain field, which is not the case for e.g. flows with fast varying mean flow. It also provides few information on the turbulence anisotropy, which would prove useful for turbulence synthesis. Reynolds stress transport models avoid this limitation by solving a partial differential equation system for the complete Reynolds stress tensor, instead of the energy k. The model is written as

$$\frac{\mathrm{D}\langle u_i u_j \rangle}{\mathrm{d}t} + \sum_k \frac{\partial}{\partial s_k}(T^{\nu}_{kij} + T^{p}_{kij} + T^{u}_{kij}) = \mathcal{P}_{ij} + \mathcal{R}_{ij} - \varepsilon_{ij} \quad . \tag{3.29}$$

The transfer tensors $T^{\nu}_{kij}, T^{p}_{kij}, T^{u}_{kij}$ describe viscous diffusion, pressure transport and turbulent convection, respectively. $\mathcal{P}_{ij}$ and ϵ_{ij} are the production and dissipation tensors, in analogy to the scalar terms introduced in § 3.3.1. The most interesting difference compared to energy transfer models is the appearance of the *redistribution term* $\mathcal{R}_{ij}$. Redistribution characterizes the transfer of energy between between isotropic and anisotropic components of turbulence. The major effect in this process is *isotropization*: Turbulence generated from the mean flow is usually highly anisotropic, and over time driven towards isotropy by energy exchange. The most commonly used realization of $\mathcal{R}_{ij}$ is the LRR-IP model [Launder et al., 1975]

$$\mathcal{R}_{ij} = -C_R \frac{\epsilon}{k}(\langle u_i u_j \rangle - \frac{2}{3} k \delta_{ij}) - C_I(\mathcal{P}_{ij} - \frac{2}{3}\mathcal{P}\delta_{ij}) \tag{3.30}$$

with the model constants $C_R = 1.8$ and $C_I = 0.6$. We will make use part of this model to augment an energy transfer model for anisotropy awareness in § 4.2.3.

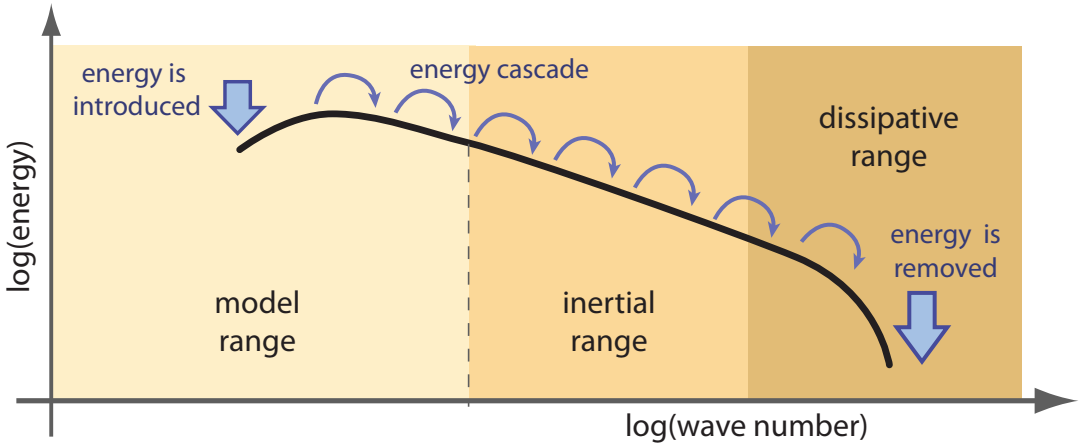


Figure 3.4: *This graph shows the typical evolution of the energy per vortex wavenumber. Energy is introduced into the system at large scales in the model range. The energy is subsequently transferred into smaller scales by scattering of vortices, and finally dissipates due to viscosity in the dissipative range.*

3.4 The Energy spectrum

In the previous sections the spatial distribution of turbulent kinetic energy was described. We will now investigate turbulence in terms of its spectral distribution. One way to approach this is the solution of the NS equation in the frequency domain, e.g. [de Frutus and Novo, 2001]. These spectral methods have some desirable properties, such as fast convergence, and drawbacks such as difficulties in boundary geometry handling. In terms of turbulence, however, they do not provide more insight than their time domain counterparts. Instead, in this chapter the spectral distribution of turbulent kinetic energy as defined in § 3.2.1 is studied.

Turbulent length scales We can think of turbulent length-scales as the size of the eddies that compose the turbulence. From experiments we can observe that while turbulent fluctuations occur on many length scales, its behavior is very different on these scales. For high-Reynolds number flows, it is observed that turbulent energy is generated mainly on large scales and dissipated on small scales. The Richardson interpretation of this states that large eddies are instable, and break up to form smaller eddies until they are eventually dissipated to heat by viscous processes[1]. This results in an *energy cascade*, that is the transfer of turbulent kinetic energy from large to small scales.

[1] *"Big whorls have little whorls, which feed on their velocity, and little whorls have lesser whorls, and so on to viscosity."* – Lewis Richardson

Following this notion, the energy spectrum can be divided into three regimes. This is also illustrated in Fig. 3.4.

- *Model range*. In this region, large-scale structures are dominant and most of the spectrum's energy is contained. Its behavior is strongly dependent on the flow geometry, and is therefore not easily described by statistical models. The production of turbulence mainly occurs in the model-dependent range by strain processes acting on the mean flow.
- *Inertial subrange*. This range shows very little production and dissipation. The main active process being forward scattering, that is the transfer of energy from small to high wave numbers.
- *Dissipation range*. The main energy dissipation occurs in the range of large wave numbers. This is driven by molecular viscosity, which is active for very small structures typically below the millimeter range.

Kolmogorov's law In his famous work, Kolmogorov [1941] proposed that for high-Reynolds number flows, fully-developed turbulence can be described very easily in a statistical sense. While large eddies are in general anisotropic and strongly influenced by boundary conditions, this behavior is lost by the energy exchange in forward scattering. Turbulence in the inertial subrange and the viscous range can thus be assumed to be *locally isotropic* and the statistical turbulent behavior in this regime is fully determined by the dissipation ε and viscosity ν. Kolmogorovs hypothesis further states that energy spectrum in the inertial subrange can be described as

$$E(\kappa) = C\varepsilon^{\frac{2}{3}}\kappa^{-\frac{5}{3}} \tag{3.31}$$

with the wavenumber κ and constant C. This is referred to as *Kolmogorovs five-thirds law* or *K41 theory*.

Beyond Kolmogorov While Kolmogorovs law is a useful tool to describe fully-developed turbulence, transition to turbulence and larger scale turbulence are not covered by K41 theory. There are, however, extensions to this model. Most notably, the energy spectrum can be extended to cover the dissipation range as well without losing too much of its generality [Pao, 1965]. This regime is not very interesting for Graphics though, as it is situated on a length scale well below the desired resolution for most Graphics simulations. The model-dependent range, on the other hand, is very hard to describe statistically in a general way. Even if such an averaged energy spectrum existed, its expressiveness would be limited, as the dynamics in the model range are dominated by anisotropic, coherent structures and their interplay with

boundary conditions. An accurate description of this regime will therefore necessarily have to track individual structures.

Another possibility of obtaining more generality is to model the energy evolution of the energy spectrum, instead of considering an stationary spectrum. This can describe some of the effects of turbulent transition. *Spectral energy transfer* models approach this in the same manner as the spatial transfer models introduced in § 3.3.1. In contrast to spatial transfer models, the individual terms are however not as easy to model, and result in complex and instable systems. Therefore, these models are mainly used to derive stationary spectra instead of transient modeling [Pope, 2000]. A review on spectral energy transfer methods can be found in [Panchev, 1971].

This being said, it is possible to describe the behavior outside K41 using additional data. We present a method that shifts the border towards the model range by explicit modeling anisotropy in § 4.2.3.

3.5 Turbulence synthesis

For CFD applications, the interest in turbulence is mainly focused on averaged properties: its influence of turbulence on the mean velocity, turbulent mixing or the induced forces. In Computer Graphics, however, the transient behavior of turbulent detail itself is important, as it makes out the desired visual appearance. Based on the observations in § 3.4 that fully-developed turbulence has a rather uniform behavior, it is possible to generate detail without a costly full simulation. This *turbulence synthesis* generates detail that obeys certain statistical properties predicted by CFD turbulence models, such as the ones described in the previous sections. Detail generated this way will therefore correspond to detail actually observed in high-resolution reference simulations or real-world experiments only in a statistical sense.

The most common approach is to synthesize a high-resolution velocity field to represent the additional detail. There are however different statistics that can be used for synthesis, and different ways to generate detail with given statistical properties. This does not mean all of these realizations will produce realistic output, though. Since turbulence generation does not imply the fulfillment of the NS equations, there is no guarantee that a given realization of a statistical representation will behave like a fluid. This means that to obtain believable results, additional information beyond the spatial distribution and the frequency spectrum are necessary. The goal of turbulence synthesis is therefore to find a combination of conditions that produces turbulent detail in a believable manner. In this section, we will introduce the commonly used

curl noise synthesis. Other synthesis approaches include random forcing [Zhao et al., 2010] or synthesis using Lagrangian vorticity primitives which will be introduced in § 5.

Believable detail To our knowledge, there is no direct comparative study on which properties are exactly required for realistic appeal of turbulence. In our experience, the qualities listed below are vital in order achieve realism and we try to obey these criteria in our methods. Recent research papers on turbulence synthesis seem to back this assessment.

1. *Solenoidality*. The most distinguished property of fluid flows is their solenoidal behavior Eq. (3.1). It is responsible for the swirly look of fluids, especially on the small scales. Thus, it is vital that the generated detail velocity field remains divergence-free in order to produce realistic results.
2. *Temporal coherence*. The temporal continuity is of equal importance as discontinuities between timesteps will cause visible artifacts in the flow field. Particular care has to be taken that coherent turbulent features such as eddies are preserved over time, otherwise turbulence appears as discoherent noise.
3. *Spectral distribution*. To obtain the characteristic look of turbulence, the distribution of vortex sizes is important. Such distribution can be for example obtained from the Kolmogorov law for the inertial subrange.
4. *Spatial distribution*. In most flows, the turbulence intensity is not homogeneous in whole domain, but areas with strong turbulence and areas with negligible turbulence will exist. This is especially true for flows with high turbulence production, such as flows around obstacles and buoyant plumes, in which turbulence intensity varies strongly. Traditionally, information about spatial distribution of turbulence is extracted from the flow field. This is however only viable for resolutions high enough to resolve turbulence generation. This thesis will introduce an approach that uses energy transfer models to predict spatial distribution, therefore allowing lower base resolutions (§ 4).

3.5.1 Curl Noise Synthesis

The most straightforward approach to satisfy both solenoidality and a prescribed spectral distribution is curl noise synthesis. In curl noise synthesis, a three-dimensional noise field N_f is synthesized from an energy spectrum. First, the desired spectrum $E(\kappa)$ is overlaid with a random phase $\varphi \in [0 \dots 1]$

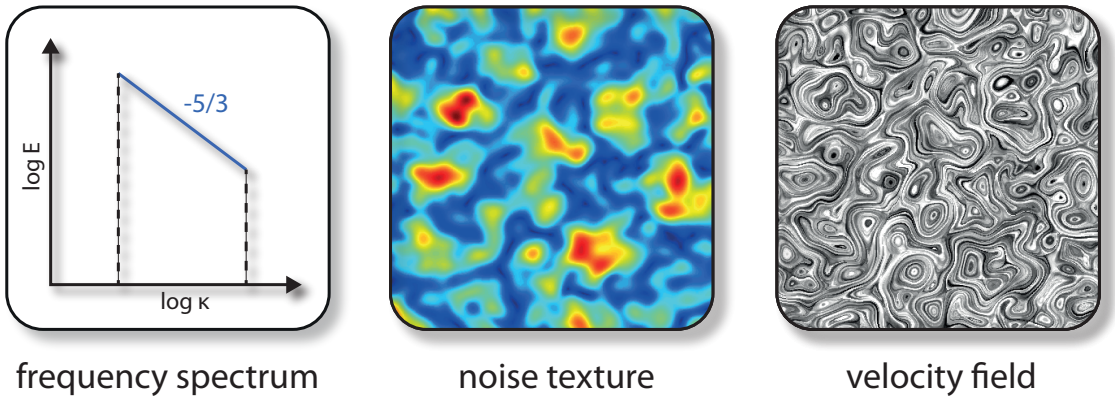


Figure 3.5: *Curl noise synthesis for the Kolmogorov spectrum. The energy spectrum is transformed into a noise field using a Fourier transform. Applying the curl operator yields a divergence-free velocity field with the desired frequency behavior.*

and transfered to a spatial noise field using the Fourier transform

$$N_f(x) = \int E(\kappa) \cdot e^{-i\kappa x + i2\pi\varphi(\kappa)} \mathrm{d}\kappa \quad . \tag{3.32}$$

The same effect can be achieved by dividing the energy spectrum into octaves, and stacking multiple octaves of a narrow-band noise field N_i, for example Wavelet Noise [Cook and DeRose, 2005], with the respective energy coefficients E_i

$$N_f(x) = \sum_i E_i N_i(x) \quad . \tag{3.33}$$

This noise field can now be used to generate a detail velocity field. First, three noise fields N_x, N_y, N_z are generated using the same energy spectrum but different phases. These fields can now be interpreted as a vector potential. The detail velocity is then generated by applying the curl operator

$$\mathbf{u}_D(\mathbf{r}) = \nabla \times \sqrt{\alpha_S} \begin{pmatrix} N_x(\mathbf{r}) \\ N_y(\mathbf{r}) \\ N_z(\mathbf{r}) \end{pmatrix} \tag{3.34}$$

with a detail strength coefficient α_S. This process is illustrated in Fig. 3.5 for the Kolmogorov spectrum Eq. (3.31). The curl operator guarantees the solenoidality of the resulting velocity field. As the curl operator is linear, the frequency characteristic of the noise field also remains intact. A detailed account on accurately computing this velocity on discrete grids can be found in Kim et al. [2008b].

3.5.2 Composition

After synthesizing a velocity field with the desired frequency spectrum, this field has to be combined with the large-scale simulation to form a coherent flow. First, a separation of the scales for model-dependent large-scale flow and uniform turbulent behavior has to be introduced. In LES, this is realized by applying frequency filters while for RANS simulations, the mean flow field $\bar{\mathbf{U}}$ represents the large-scale simulation. Methods employed in Graphics use a notion similar to RANS. As base simulations with low resolution and a diffusive semi-Lagrangrian advection typically have an inherent diffusion higher than the turbulent diffusion in RANS, the simulation is used directly as the large-scale flow. For synthesis methods based on K41, this assumes the grid resolution marks the division between the model-dependent range and the inertial subrange. The former is then represented by the large-scale flow, while later is obtained by synthesized sub-grid detail.

The most common solution to store sub-grid detail is using a grid of higher resolution than the base simulation. Some methods also operate on the same grid resolution as the base simulation – this makes sense as the frequency cutoff induced by numerical dissipation happens on the scale of the multiple grid cells. However, the improvement in detail obtained this way is obviously limited.

To combine large-scale flow and synthetic detail, these two fields are composed. As a first step, this requires upsampling the large-scale velocity field to the resolution of the detail field. If the frequency spectra of the two fields can be assumed to be disjunct in the sense of RANS, they can be simply added (Fig. 3.6). If, on the other hand, the spectra overlap, care has to be taken that features are not duplicated. This is especially the case if detail field and large-scale flow fields are of the same resolution. In this case, reinforcement techniques are applied: The turbulence intensity is measured on the large-scale field, and compared to the detail field. Only the difference between these fields is then added to form the resulting field. This approach is used e.g. in vorticity reinforcement [Selle et al., 2005] and vorticity confinement [Fedkiw et al., 2001].

Spatial distribution Combining a simulation with a synthesized detail field as above results in uniform, homogeneous turbulence over the complete field. For most scenarios, this is not sufficient, as turbulence intensity will vary over the domain. Here, *turbulence predictors* are employed to estimate the spatial distribution of turbulence strength. Simple turbulence predictors measure small-scale whirls present in the base simulation by vorticity [Fedkiw et al.,

Figure 3.6: *A large-scale flow field from a simulation is combined with detail generated by curl noise synthesis.*

2001] or wavelet decomposition [Kim et al., 2008b] and assume they form the upper level of the turbulent energy cascade. Based on their energy, the lower levels of turbulence can then be reconstructed. This however requires a simulation resolution high enough that turbulence is formed at all. In our methods, we employ energy transfer models as described in § 3.3.1 to estimate the spatial turbulence intensity.

To incorporate spatially varying fields of turbulence intensity obtained either way into curl noise synthesis, the detail strength coefficient in Eq. (3.34) can be modified. Based on Eq. (3.31), this can be achieved either via the turbulent kinetic energy or dissipation

$$\alpha_S(\mathbf{r}) \propto k \propto \varepsilon^{\frac{2}{3}} \quad . \tag{3.35}$$

Strictly speaking, this violates solenoidality, as the modulation may introduce divergence. In practice, this is not a problem as long as the gradient of α_S is not too steep, as these divergences do not accumulate over time. For steep gradients, such as interfaces of buoyant plumes, this will however lead to visual artifacts. A more correct synthesis could be achieved by incorporating the spatial intensity distribution directly into the synthesis, for example using wavelets. To our knowledge, this approach has not been used in any synthesis method so far, which is largely due to the numerical complexity of performing a full spectral transfer in each simulation frame.

Temporal coherence One of the most intricate issues in turbulence synthesis is ensuring temporal coherence. This is due to the fact that the two main goals in temporal coherence are incompatible. First, the generated velocity field should deform in accordance with the flow. As the turbulent energy cascade mainly involves forward scattering, it is assumed that the detail field

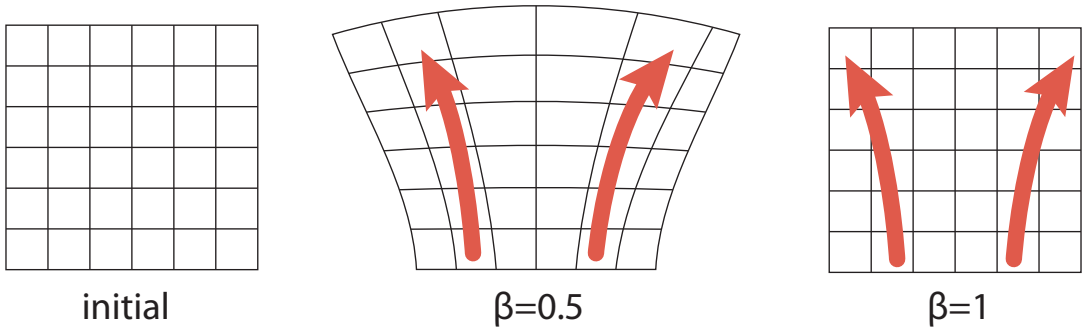


Figure 3.7: *The texture coordinate field t_2 is depicted. Initially, the coordinates correspond to position in space. Over time, the field deforms due to mean flow velocity, drawn in red here. The field t_2 is reset on $\beta = 1$, when its coefficient in Eq. (3.36) is zero.*

remains passive, and moves within the large-scale flow. It should however also not strongly deform, as this will distort the frequency behavior, and destroy the characteristic turbulent shapes. By advecting the detail field in the large-scale flow, deformations accumulate and will inevitably induce strong deformation. This is also a common problem in texture synthesis, and the solutions are similar in both fields.

The advection in the large-scale flow can be realized using texture coordinates which index positions in the detail field. At the start of the simulation, the texture coordinates will correspond to their position in space. Over the course of the simulation, they are advected in the velocity field which leads to distortion of the field. The most commonly used technique in preventing strong distortions is coordinate resetting. After a number of steps, all coordinates are reset to their position in space. As this will naturally induce jumps in the velocity field, two sets of texture coordinates are used, and reset alternately. The generated velocity at a point is then the linear combination based on its two texture coordinates t_1, t_2.

$$\mathbf{u}' = \beta \mathbf{u}_D(t_1) + (1-\beta)\mathbf{u}_D(t_2) \tag{3.36}$$

with $\beta \in [0,1]$ being a sawtooth function in time. The texture coordinate sets can be reset when its respective coefficient is zero. This process is illustrated in Fig. 3.7. There are also alternative approaches, such as local resetting based on deformation strength [Kim et al., 2008b].

3.5.3 Discussion

When applying turbulence methods, it is important to realize the limits of the used model and statistic methods in general. This is a point often neglected in turbulence methods for Graphics. Some violations of the limits will not be visible and may be tolerable, due to the fact that the human perception system is not trained to spot inaccuracies in fluid dynamics. Others might severely affect the perceived realism of the scene. This will also heavily depend on scene setup and rendering – for example, the anisotropic turbulent transition region is directly visible for dense smoke, while its effect is less visible for diffuse smoke. Therefore, statistical, isotropic turbulence models as the ones described in the previous chapters are likely to produce artifacts for dense smoke clouds, while results may perceived realistically for a similar setup with diffuse smoke. In absence of solid perceptional studies, it is therefore best to be clear about the limits and its violations of the model used. Below, some common pitfalls and limits of turbulence methods are listed.

The Scales of Turbulence Turbulence modeling and synthesis base on the fact that turbulence can be separated from the mean flow, and has a uniform dynamic that is well-described by statistical properties. This is true only under certain conditions. Most importantly, turbulence should only be generated for the inertial subrange. Larger scales show nontrivial interaction with flow obstacles, and both forward and backward scattering. This means not only is the K41 energy distribution not valid in this regime, but the dynamics are dominated by coherent structures that can hardly be captured by a statistical model. Here, a simulation is essential as synthesized turbulence will inevitably introduce an unnatural look. This means that the division between simulation resolution and generated subgrid detail has to be carefully chosen. Another aspect of this is that turbulence models base their prediction on the mean flow. This means the predicted turbulence intensity will change depending on the base simulation resolution. This is especially critical if many turbulent details are already resolved by the base solver, as the details will act as turbulence sources, resulting in a strong overprediction of turbulence. In these cases, a full RANS simulation, or an averaged flow field should be used instead of the base simulation.

Isotropy Both the turbulence models and synthesis algorithms presented in this chapter only take into account isotropic turbulence. This is a good assumption for fully developed turbulent flows, but not for areas of turbulence generation or transition. This is due to the fact that turbulence generated from

shear will create whirls with a preferred direction, which will only become isotropic over time (see § 3.3.2). In flows where these areas are clearly visible, such as open channels and turbulent dense smoke, isotropic models should not be used – isotropic turbulence will be perceived as noise disturbing the flow. To include the effects of anisotropy, extensions to both turbulence prediction and synthesis have to be made. Such a model is presented in § 4.2.3. However, even with anisotropic methods, such as Reynolds stress transport models, there is no guarantee that turbulence transition is well represented. Turbulence induced by breakdown for example from large coherent structures can hardly be represented using a statistical approach – for this, the breakdown has to be modeled explicitly with methods such as the one presented in § 6.

Accuracy of Synthesis Synthesis methods have to fulfill several constraints, which are often incompatible, as discussed in § 3.5.1. Therefore inevitably compromise solutions have to be implemented, whose effectiveness will depend on a good choice of parameters. Synthesis based on curl noise is especially problematic at steep interfaces of turbulence intensity, e.g. buoyant smoke with a sharp density gradient, and flows with strong strain effects. The latter will induce either strong deformations of the turbulence field, or interpolation artifacts associated with frequent resets. Either way, coherent whirls may be reduced to structure-less noise.

Liquids Although in principle turbulence modeling can be applied to liquids as well as smoke and fire, several issues prevent an efficient adoption of turbulence methods for liquids. First, the energy transfer near the liquid surface is not well-described by turbulence models. This is a topic of ongoing research in the CFD community, so far no standard theory as universal and accurate as bulk flow turbulence theory has been established. The synthesis step is even more problematic for liquids. The most interesting region is the fluid surface, which is shown in rendering. However, turbulence at the interface has a completely different dynamic in the bulk flow. The reason for this is that the interface breaks the separation of mean flow and turbulent detail. If for example a whirl is synthesized on the surface, secondary waves will form, which may influence the mean flow. This means applying synthesized subgrid turbulence is not sufficient, but subgrid interface dynamic has to be simulated as well, which nullifies the performance gain of the turbulence method. It is however possible to use turbulence reinforcement in the bulk flow, which will not increase resolution but counteract unwanted turbulent dissipation in the flow.

Outlook In this chapter, the state-of-the art of classical turbulence modeling and vorticity-based fluid simulation was summarized. In the following chapters, we will present the methods developed during the course of this PhD thesis. We build on the theory presented here, extend and adapt the models to the task of enhancing realism in fluid simulations for Computer Graphics. First, a general purpose turbulence method for predicting and synthesizing turbulent detail is presented in § 4. Our approach is geared towards interactive applications, and designed to be able to produce fine detail at real time. In § 5 we will then investigate the formation of turbulence in the wake of obstacles, and develop a method to directly model these formation processes which allows for realistic detail in the transition range. Finally, we will study interface effects of turbulent buoyant plumes in § 6. By appying vortex methods directly on the interface of the plume, we can generate detail on the surface without the need of volumetric computations.

CHAPTER 4

Real-Time Turbulence Methods

In this chapter, we will develop methods to simulate highly detailed, turbulent fluid systems in real time. To this end, we will investigate turbulence synthesis and prediction techniques which are able to represent complex features of turbulence without strong reliance on the base solver, and which are suitable for GPGPU computation. Such approaches will prove useful to enhance the realism of fluid simulations in interactive applications, such as Computer Games.

It is generally very difficult to resolve the fine details of turbulent flows in a simulation, and real-time systems impose even stricter limits on the simulation resolution used. Real-time simulations would therefore be a prime field of application for turbulence synthesis methods – as synthesizing turbulent detail is more efficient and scales better than the direct simulation of this detail. However, turbulence prediction using real-time solvers is non-trivial. The low resolution available to the base simulation will result in a coarse and diffusive velocity field which dampens out flow instabilities. This means many types of turbulence will not even be generated in this base simulation. Methods such as the popular Wavelet Turbulence [Kim et al., 2008b] which directly rely on the base grid as a turbulence predictor will therefore fail in this scenario. In addition, the real-time constraint precludes the use of high-resolution grids to store and render the generated turbulence.

We therefore present a synthesis method which uses an energy transfer model

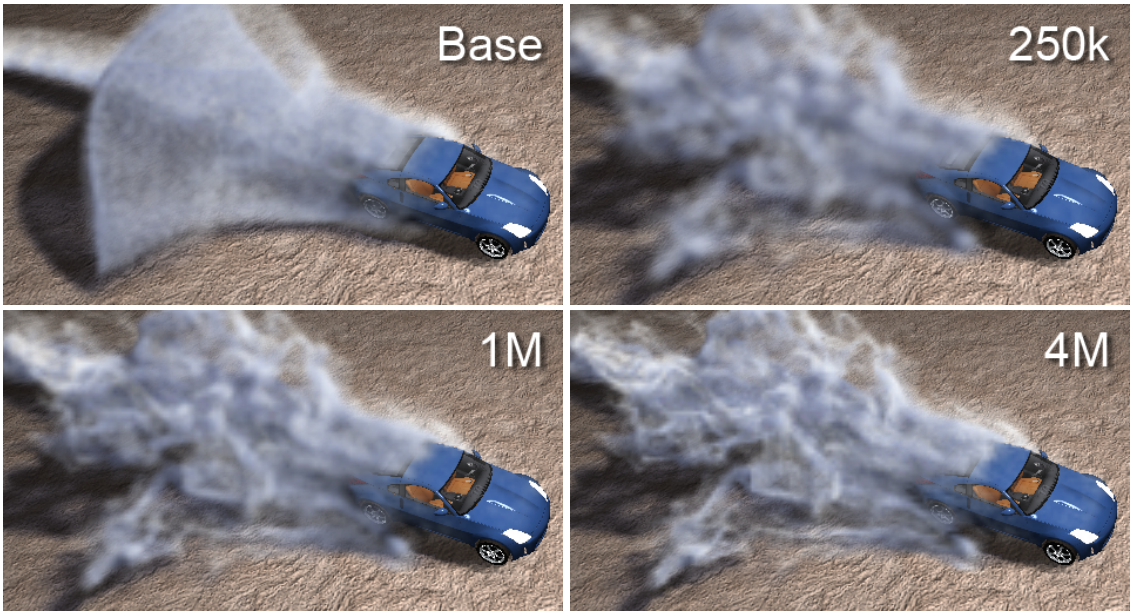


Figure 4.1: *Here, the simulation of the wake of a moving car is shown. The base simulation in the top left pictures uses a resolution of only* $32 \times 8 \times 32$*. This simulation is augmented with our turbulence method, with a varying number of particles from 250k to 1M and 4M from left to right. For the simulation with 1M particles we achieve 15 frames per second on average, including rendering. While the amount of detail directly depends on the number of particles used, the overall flow remains consistent.*

to correctly predict turbulence intensity even on low resolution grids. In particular, we explicitly model anisotropy, which allows us the represent the anisotropic turbulence formation process using turbulence synthesis, instead of relying on the simulation, enabling realistic behavior even at very low base resolutions. The method is designed to operate directly on the Lagrangian markers used for rendering, which eliminates the need for a costly high-resolution grid representation and allows us to perform synthesis exactly where needed. As we offload complexity from the fluid solver to the particle system, we can control the detail of the simulation easily by adjusting the number of particles, without changing the large scale behavior. We demonstrate that our algorithm is highly suitable for massively parallel architectures, and is able to generate detailed turbulent simulations with millions of particles at high frame-rates.

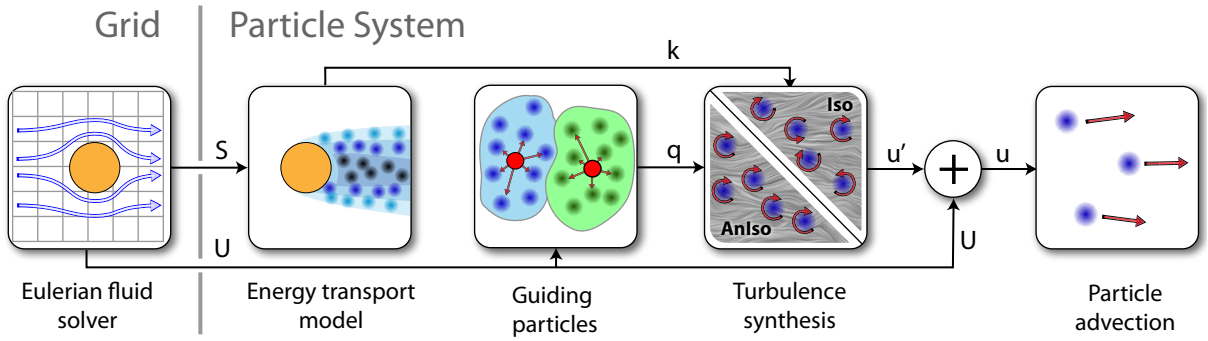


Figure 4.2: *An overview of our algorithm. A low resolution grid-based solver provides a base velocity and strain field. For each particle, a turbulence model is computed, which drives the turbulence synthesis with isotropic and anisotropic turbulence. The particles velocity is given by the large scale velocity from the grid and the small scale turbulent velocity.*

4.1 Overview

To efficiently simulate small scale turbulence it is not feasible to directly represent the turbulent motion using a high resolution velocity grid, as the computational effort increases strongly with grid resolution. Instead, we describe the turbulence field by its statistical properties, and synthesize turbulence only where needed. Our approach is based on a separation of the large scale dynamics from the small scale turbulence: large scales are computed using a low resolution fluid solver, while the turbulent detail with anisotropic effects is computed on the particle system. The particles coincide with the smoke particles used for rendering, while the grid-based solver is used to obtain the large scale characteristics of the flow. The turbulence is computed and synthesized directly on the particles, each of which is influenced by a texture based turbulence representation, and stores a preferred axis of rotation for anisotropic effects. An overview of our model is given in Fig. 4.2.

In § 4.2 we describe the modified energy transport model used for prediction of the spatial distribution of turbulence. This drives the synthesis model § 4.2.2, which is extended for anisotropy in § 4.2.3. § 4.3 discusses implementation details of our approach. Finally, the complete model is tested in reference scenarios and compared to other methods in § 4.4.

4.2 Turbulence Model

We chose to describe the turbulence using an energy representation, as this allows us to adapt powerful transport models for our simulations. The spatial and temporal energy distributions driving the turbulence synthesis are obtained using a modified k–ε turbulence model that we will explain in the following sections.

4.2.1 Energy transport

To simulate the energy dynamics, we use a modified version of the k–ε model by Launder and Sharma [1974]), which is one of the most widely used turbulence models in CFD. It is a complete two-equation model, which unlike one-equation models used by previous methods requires no additional problem-dependent assumptions. On the basis of a large scale flow field $\bar{\mathbf{U}}$, it models the two variables k and ε on an averaged large scale. While k represents the turbulent kinetic energy contained in the smaller scales, ε stands for the dissipation of the turbulence structures. The k–ε model and other turbulence models are discussed in the theory section § 3.3.1.

We start with the partial differential modeling equations of the k–ε model

$$\frac{\partial k}{\partial t} + \bar{\mathbf{U}} \nabla k = \nabla \left(\frac{\nu_T}{\sigma_1} \nabla k \right) + P - \varepsilon \tag{4.1}$$

$$\frac{\partial \varepsilon}{\partial t} + \bar{\mathbf{U}} \nabla \varepsilon = \nabla \left(\frac{\nu_T}{\sigma_2} \nabla \varepsilon \right) + \frac{\varepsilon}{k} \left(C_1 P - C_2 \varepsilon \right) . \tag{4.2}$$

Both equations share the same structure: the left-hand side contains an advection in the mean flow field. The right-hand side of both equations consist of a viscous diffusion term, a production and a dissipation term, in that order. The equations are coupled to the mean flow field via the velocity field $\bar{\mathbf{U}}$ and the mean flow strain S_{ij} in the production term

$$P = 2\nu_T \sum_{ij} {S_{ij}}^2 . \tag{4.3}$$

Instead of discretizing and solving these PDEs on a Eulerian grid, we compute them directly on the particle system. This means each particle stores a value of k and ε, while the coupling parameters of mean flow velocity and mean flow strain are interpolated from the base flow field. In this Lagrangian setting, Eq. (4.1) and Eq. (4.2) simplify, as the advection is inherently handled by the motion of the particles with the flow. Next, let us reconsider the role of the

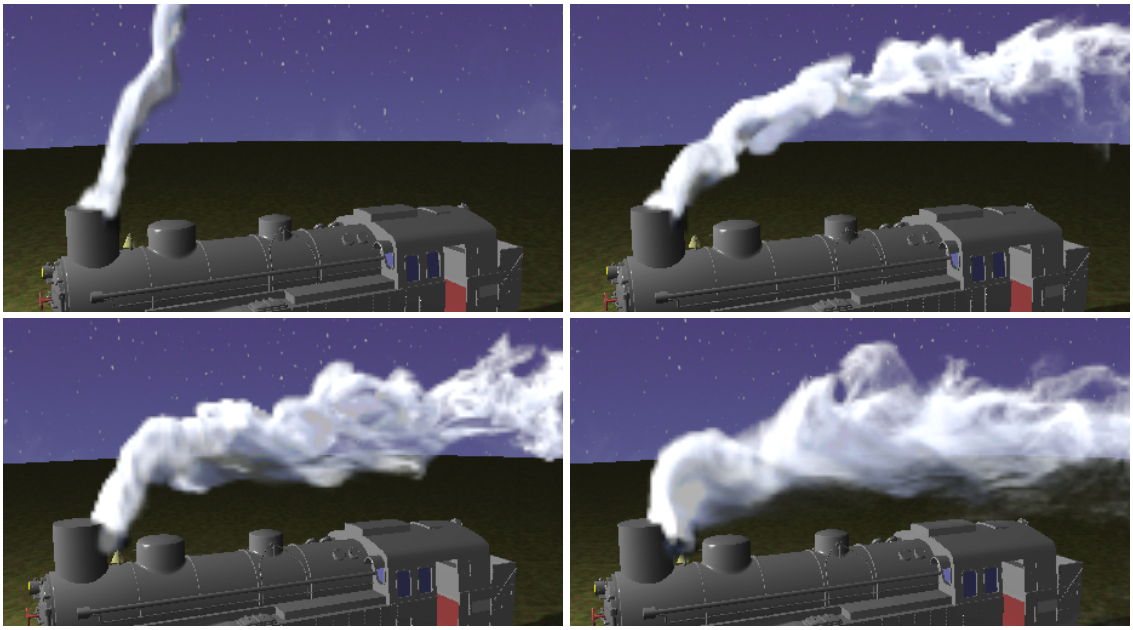


Figure 4.3: *We apply our method to an accelerating train that, in the end, comes to an abrupt halt. Due to our energy transport model, turbulence intensities correctly adapt to the direction of the flow and the train's velocity.*

diffusion term. The turbulent viscosity ν_T is a virtual diffusion, which models the averaged effect of the small-scale turbulent motion as a viscous diffusive effect on the larger scale of the model. In contrast to CFD however, we solve the model equations and track our turbulence properties on the particle set. The particle set is not only advected by the averaged large-scale flow, but also contains a secondary advection by the synthesized detail motion. This small-scale advection term causes turbulent mixing of the particles and thus implicitly describes the behavior which is modeled by turbulent diffusion term for the large scales in the standard k–ε model. Therefore, the turbulent viscosity term vanishes in our representation. Avoiding these terms also allows us to track k and ε independently for all particles and skip a costly communication step, which is important for GPGPU parallelization. So far, the isotropic version of our model therefore consists of the following equations

$$\frac{\mathrm{D}k}{\mathrm{D}t} = P - \varepsilon \tag{4.4}$$

$$\frac{\mathrm{D}\varepsilon}{\mathrm{D}t} = \frac{\varepsilon}{k}(C_1 P - C_2 \varepsilon). \tag{4.5}$$

4.2.2 Turbulence Synthesis

To obtain the detailed motion for the particle system, in addition to the base flow velocity, a detail velocity component is evaluated directly on the particles. Unlike previous works which rely on the base solver to predict the turbulence distribution, we will use the energy transfer model described in the previous chapter to estimate the spatial distribution of turbulence. We will discuss our energy model spectrum, and then describe our approach to compute isotropic turbulence. Extensions for anisotropic effects will be presented in the following chapter.

Model Spectrum Turbulent energy is usually studied with respect to the spatial scales of the structures in the flow field. The production of turbulent energy is typically concentrated in the energy-containing range of a fluid, its large scales, while dissipation to heat is growing stronger for the small scales. In between these two extrema lies the so called inertial subrange, in which the predominant energy transport mechanism is forward-scattering, transporting the energy from large to small scales. This transfer process can be modeled with time dependence, e.g. using the transient model of Obukhov [1941]. However, the model is often not practical, as the exponential nature of the transfer terms requires a high spectral and temporal resolution for a stable solution [Panchev, 1971]. Fortunately the transfer phenomena in a flow quickly drive the distribution to a stationary solution for fully-developed turbulence. Therefore, practical approaches typically focus on the stationary solution only. Our method concentrates on scales mostly within the inertial subrange for which the well known Kolmogorov K41 law, as in [Frisch, 1995], is a reasonable approximation.

Synthesis To synthesize isotropic turbulence, frequency-matched curl noise (§ 3.5.1) is used. Similar to [Kim et al., 2008b] the spectrum is divided into N octaves and each band is synthesized using the curl of band-limited wavelet noise [Cook and DeRose, 2005] with band coefficients determined using the K41 law. In all our demos, we uses three octaves. The total velocity $\mathbf{u}$ of a particle is then given by the large scale flow velocity $\bar{\mathbf{U}}$, interpolated from the low resolution Eulerian solver, and the turbulence velocity:

$$\mathbf{u} = \bar{\mathbf{U}} + 2(\alpha_S k)^{\frac{1}{2}} \sum_{i}^{N} \mathbf{c}_i(\mathbf{q})\, 2^{-\frac{5}{6}i} \tag{4.6}$$

Here, $\mathbf{c}_i(\mathbf{q}) = \nabla \times \mathbf{N}_i(\mathbf{q})$ are the curl noise textures as described in § 3.5.1. $\mathbf{q}$ represents a texture coordinate, and k denotes the energy of the largest-scale

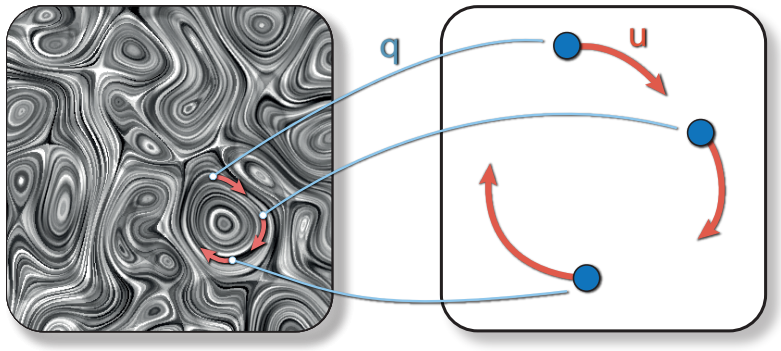


Figure 4.4: *Each particle owns a lookup coordinate into the turbulence velocity texture. Both the particle position and the texture coordinate are advected by this velocity, allowing coherent eddies to form.*

turbulence band. Instead of storing the texture coordinate on a grid as in Kim et al. [2008b], in our case the coordinate is stored directly on the particle system. As we only model forward scattering, we can assume the turbulent detail map is passively advected in the large-scale flow. This means that both particles position and the texture coordinate are updated using the turbulent detail velocity, but only the particle position is advected using the large-scale flow velocity $\bar{\mathbf{U}}$. This allows the formation of coherent turbulent whirls over multiple particles, as shown in Fig. 4.4.

The scaling of the wavelet turbulence is chosen such that the largest synthesized vortices cover 2–4 grid cells, as vortices on these scales are usually dampened out by numerical viscosity of the Eulerian simulation which is better able to represent larger vortices. The energy for the synthetic turbulence is directly given by the $k-\varepsilon$ model with $k = k_{iso} + k_{an}$ for each particle, where k_{iso} denotes the isotropic energy, and k_{an} the anisotropic energy. Assuming $k_{an} = 0$ for now, the energy for the largest band is given by $\alpha\, k_{iso}$. The scaling parameter α encodes the shape of the assumed energy spectrum, and can be used to artificially increase or decrease the strength of the turbulence. This is one of the two main tuning parameters of our model.

4.2.3 Anisotropy

So far we have only considered isotropic turbulence. Some important effects, however, most notably the production of turbulence, are highly anisotropic. Neglecting anisotropy would therefore result in turbulence structures that are not fully connected to the motion of the underlying base flow. Capturing anisotropy requires both a way to synthesize anisotropic noise, as well

as an energy transport model capable of providing the anisotropic energy distribution. For the latter, Reynolds stress transport models as mentioned in § 3.3.2 can be used. These models describe the evolution of tensor quantities, most notably the Reynolds stress tensor. While the complete model is far too complex to be applied in real-time applications, we will use selected elements of this theory to augment our model. According to [Pope, 2000], the most relevant case of anisotropy is the production of elongated vortex structures due to shear effects. This happens, e.g., at boundaries and leads to rotational velocities perpendicular to the shear plane, reducing the dimensionality of the effect from three to two dimensions. Therefore, we consider the case of turbulence consisting of an isotropic component with energy k_{iso} that is handled as described above, and a completely anisotropic two-dimensional component with the energy vector $\mathbf{k_A}$. The direction of $\mathbf{k_A}$ defines the normal of a plane to which the anisotropic turbulence is confined. This is equivalent to a preferred rotation axis, and the magnitude of $\mathbf{k_A}$ defines the energy contained in the anisotropic vortices.

Energy Dynamics While the k–ε equations Eq. (4.4),(4.5) still hold for the total energy $k = k_{iso} + k_{an} = k_{iso} + |\mathbf{k_A}|$, we need to determine the evolution equation of the anisotropic component $\mathbf{k_A}$. The mechanisms here are similar to transport of total energy: there is a production, a dissipation and additionally, a redistribution term. Analogous to Eq. (4.3), the production vector is given by the turbulent viscosity ν_T and the strain. We use an eigendecomposition to divide the strain tensor into an anisotropic component, represented with two-dimensional turbulence, and an isotropic component. In the following, λ_i denote the eigenvalues and $\mathbf{v_i}$ the eigenvectors of S_{ij}, where λ_1 has the biggest and λ_3 the smallest absolute value. Now consider the production ellipsoid defined by the vectors $\mathbf{p_i} = 2\nu_T \lambda_i^2 \mathbf{v_i}$. The plane of two-dimensional shear stress is spanned by its two longest vectors $\mathbf{p_1}$, $\mathbf{p_2}$, while the plane normal is given by $\mathbf{v_3}$. The isotropic component, on the other hand, is the sphere spanned by the smallest common component of all vectors, that is $|\mathbf{p_3}|$. Therefore, we can define the anisotropic production vector to be

$$\mathbf{P_A} = 2\nu_T (\lambda_1^2 + \lambda_2^2 - 2\lambda_3^2)\, \mathbf{v_3} \quad . \tag{4.7}$$

While in areas of high production, e.g. near obstacle boundaries, anisotropic effects can be observed, the turbulence further away from these regions is largely isotropic. This is due to the fact that transport processes lead to a quick isotropization of the turbulent flow. This is true for spatial turbulent transport as well for transport through the energy cascade. The LRR-IP model mentioned in § 3.3.2 models this isotropization. If we transfer this model to

our turbulence setting, it yields an energy transfer rate of

$$\frac{\mathrm{D}\mathbf{k_A}}{\mathrm{D}t} = (1 - C_A)\mathbf{P_A} - C_R \frac{\varepsilon}{k}\mathbf{k_A}\,. \tag{4.8}$$

from $|\mathbf{k_A}|$ to k_{iso}. The standard model constants are defined as $C_A = 0.6$, $C_R = 1.8$. Here, the dissipation ε is generally assumed to be isotropic, as it occurs on very small scales, while the anisotropic vortices are initiated primarily on the larger scales. This means that it is sufficient to solve the isotropic Eq. (4.5) for dissipation.

Synthesis We now extend the synthesis algorithm from § 4.2.2 for anisotropy by including additional turbulence bands. As the turbulent kinetic energy k is composed of an isotropic component k_{iso} and an anisotropic component $\mathbf{k}_A$, we will compose the synthesis term using an isotropic and an anisotropic part. The isotropic part is equivalent to Eq. (4.6) with k_{iso} instead of the total energy k. For the anisotropic component, on the other hand, the 2D curl noise field

$$\mathbf{c}^{2D}(\mathbf{q}, \mathbf{k_A}) = \nabla \times N(\mathbf{q})\,\frac{\mathbf{k_A}}{|\mathbf{k_A}|} \tag{4.9}$$

is used. It is aligned to $\mathbf{k}_A$, and thus generates turbulent eddies in the plane normal to the anisotropy vector. For easier precomputation, we effectively use the field $\mathbf{c}^{2D}(\mathbf{q}) = \mathbf{c}^{2D}(\mathbf{q}, \mathbf{e}_z)$ and then apply the rotation operator $\mathbf{R}(\mathbf{k}_A) = \mathrm{Rot}(\mathbf{e}_z \leftarrow \mathbf{k}_A)$ during the lookup. The total velocity $\mathbf{u}$ of a particle can therefore be determined by the equation

$$\begin{aligned}\mathbf{u} = \bar{\mathbf{U}} + 2(\alpha k_{iso})^{\frac{1}{2}} \sum_i^N \mathbf{c}_i(\mathbf{q})\, 2^{-\frac{5}{6}i} + \\ 2\,|\alpha\,\mathbf{k_A}|^{\frac{1}{2}} \sum_j^M \mathbf{R}(\mathbf{k_A})\,\mathbf{c}_j^{2D}(\mathbf{q})\, 2^{-\frac{5}{6}j}\end{aligned} \tag{4.10}$$

with $k_{iso} = k - |\mathbf{k_A}|$. As anisotropy decays quickly in the spectral cascade, we have found that it is sufficient to use one band of anisotropic turbulence for the largest scale.

4.3 Implementation

We have implemented our model to execute both the Eulerian fluid simulation and the particle based turbulence model on a GPU. For the underlying Eulerian solver, we use a typical MAC discretization with second order semi-Lagrangian advection, as described in [Bridson, 2008] and [Selle et al., 2008].

Our implementation makes use of a multi-grid solver for computing the pressure correction, as described in [Cohen et al., 2010]. For the Lagrangian turbulence model, each particle stores its position and velocity as well as the turbulence parameters $k, \varepsilon, \mathbf{k_A}$ and $\mathbf{q}$. The evolution of these variables is given by integrating Eq. (4.4), Eq. (4.5), Eq. (4.8), and evaluating Eq. (4.10), respectively, on the particle system. We use a simple forward Euler integrator for all of these equations. The strain eigen-decomposition required for Eq. (4.7) is calculated on each grid cell. As the 3×3 strain tensor is symmetric, eigenvalues can be found efficiently using the analytic formulation [Smith, 1961]. Our model is designed to work without any particle-particle interactions, and only a few linear interpolations of data from simulation grid for velocity and strain are necessary to compute the particle dynamics. This makes it very efficient to compute even in massively parallel settings. In our setup, the smoke is rendered online using half-angle volumetric shadowing by Ikits et al. [2004], enabling the complete framework to run at interactive frame-rates and therefore providing immediate results. Below, we will discuss important details concerning stability and initialization.

Stability The k–ε model, being a coupled system of two PDEs in its original form, has inherent stability problems. Especially k in the denominator of Eq. (4.5) causes instabilities for flows with low turbulence. Therefore, the model is usually modified to guarantee stability. A commonly used approach is a low-Reynolds number treatment, as described in [Pope, 2000]. We use a simplified version of this approach to ensure that a minimal turbulent energy is always present in the simulation.

A meaningful range for the turbulent energy k is given by $k = \frac{3}{2} {U_0}^2 I^2$, with the turbulent intensity $I \in [0 \dots 1]$. Here, U_0 is the characteristic velocity scale, which can be determined from the simulation parameters, e.g., the maximal speed of the car in Fig. 4.1. As suggested in the field of aerodynamics research [Spalart and Rumsey, 2007], we use a value of $I_{min} = 10^{-3}$ as a minimal turbulent intensity, while, naturally, the maximal intensity is given by $I_{max} = 1$. By restricting the simulation to this meaningful range of values, the system quickly recovers from overshoots and is stable for arbitrary time steps.

Similarly, we can define a corresponding range for the values of ε. We obtain a minimum dissipation by specifying a minimal turbulent viscosity equal to the molecular viscosity of air ν_{air}, which represents a natural lower bound for the viscosity of smoke simulations. Using the definition of turbulent viscosity Eq. (3.26) for the k–ε model, we obtain $\varepsilon_{min} = C_\mu \frac{{k_{min}}^2}{\nu_{air}}$. The maximal dissipation, on the other hand, can be derived on the basis of a minimal

turbulent length scale L_{min}, and is given by $\varepsilon_{max} = C_\mu{}^{\frac{3}{4}} k_{max}{}^{\frac{3}{2}} \frac{1}{L_{min}}$. We use a minimal length scale of $\frac{1}{10}$ of a grid cell in our simulations.

Note that these ranges for turbulent energy and dissipation are useful when allowing users to interact with the simulation. They can, e.g., provide artists with intuitive parameter ranges for setting up turbulence sources in a scene.

Initial state We seed the particles at the smoke inflow of the scene. As this will usually not coincide with the fluid inflow region, we need to specify sensible initial values for the turbulence parameters of these particles. In cases where the inlet is in a low-turbulence area, we can use the lower boundaries k_0 and ε_0 as initial values. If, on the other hand, the smoke should be generated in a turbulent region, we need to specify initial energy levels, as we have no information about the history of the particles. This can be achieved with different approaches. We estimate typical turbulent intensities for k and ε similar to the estimation of maximal bounds described in the previous paragraph, and use these values for initializing the particles. Here, the minimal length scale L_{inlet} is another important parameter of our model, and can be used to tune the amount of turbulence injected into the scene. Another possibility is to initialize particles with the lower bounds k_0 and ε_0, and then perform a small number of iterations of the turbulence model on the newly seeded particles.

Texture Advection Naturally, the structure of the turbulence should deform as given by the motion of the flow. However, using a naive approach, e.g., updating the local texture coordinate $\mathbf{q}$ of each particle using only the large scale motion with $\frac{\mathrm{D}\mathbf{q}}{\mathrm{D}t} = \mathbf{u} - \bar{\mathbf{U}}$ quickly destroys coherence of the turbulent structures. By compression and mixing in flow, adjacent particles will eventually own strongly divergent texture coordinates. This destroys coherent structures as in Fig. 4.4 and will inevitably lead to uniform noise instead of recognizable swirling motions. This loss of coherence is closely related to the problem of texture field deformation in methods such as [Kim et al., 2008b]. We however do not want to rely on local resetting, as by construction, our aim is to update each particle without having to know about its neighbors. It is therefore undesirable to perform any kind of spatial interpolation on the particles.

We instead use *guiding particles* to preserve the local coherence of the turbulence. Guiding particles are seeded together with the actual smoke particles, and assigned to a small group of smoke particles based on local neighborhood. On seeding, each guiding particle is assigned a fixed texture coordinate

1: *// Grid-based Fluid solver*
2: Semi-Lagrangian advection of $\bar{\mathbf{U}}$
3: Pressure projection
4: Calculate strain field S_{ij}
5:
6: Seed and initialize new particles
7:
8: **for each** particle **do**
9: Sample U, S_{ij} at particle position $\mathbf{x}$
10:
11: *// Energy dynamics*
12: Compute turbulent viscosity: $\nu_T \leftarrow C_\mu \frac{k^2}{\varepsilon}$
13: Compute production: $\mathbf{P} \leftarrow$Eq. (4.7)
14: Integrate $k \leftarrow k + \Delta t\,(|\mathbf{P}| - \varepsilon)$
15: Integrate $\varepsilon \leftarrow \varepsilon + \Delta t \frac{\varepsilon}{k}(C_1 P - C_2 \varepsilon)$
16: Energy transfer: $\mathbf{k_A} \leftarrow \mathbf{k_A} + \Delta t\,(1 - C_2)\mathbf{P} - \Delta t\, C_R \frac{\varepsilon}{k}\mathbf{k_A}$
17: Stabilize k, ε using $k_{min,max}$ and $\varepsilon_{min,max}$
18:
19: *// Motion equations*
20: Synthesize velocity: $\mathbf{u} \leftarrow$ Eq. (4.10)
21: Integrate $\mathbf{x} \leftarrow \mathbf{x} + \Delta t\,\mathbf{u}$
22: Integrate $\mathbf{q} \leftarrow \mathbf{q} + \Delta t\,(\mathbf{q_G} + \mathbf{x} - \mathbf{x_G})$
23: **end for**
24: Advect guiding particles in flow field $\bar{\mathbf{U}}$
25:
26: Render simulation data

Figure 4.5: *Pseudo-code for the simulation loop.*

$\mathbf{q_G}$ based on its world coordinate, which acts as a local frame of reference for the texture coordinates of the attached smoke particles (Fig. 4.6).

As the guiding particles represent the motion of the turbulence textures, they are advected using only the large scale flow from the underlying simulation. The local texture coordinate of each smoke particle can now be computed using the local position $\mathbf{x}$ with respect to the assigned guiding particle as $\mathbf{q} = \mathbf{x} - \mathbf{x_G} + \mathbf{q_G}$. This approach allows us to efficiently preserve locality while adhering the turbulence motion to the large scale flow. While coherence and incompressibility are exactly preserved within the particle cloud of a guiding particle, coherence loss and small-scale deviations from incompressibility may appear between these clouds. Therefore, guiding particles should be

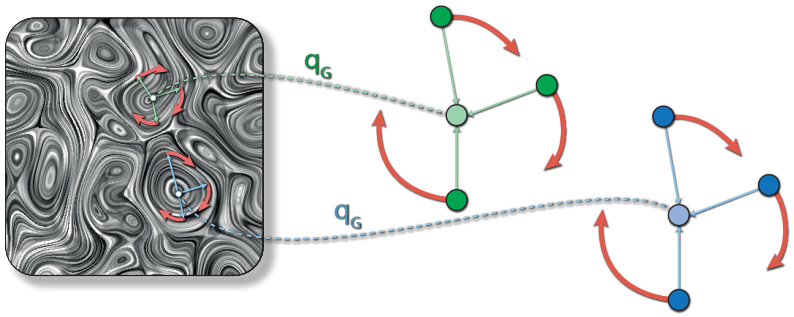


Figure 4.6: *Two groups of particles (blue, green) with an associated guiding particle are shown. The texture coordinate lookup of the individual particles is performed by taking the geometric distance to the guiding particle, and the associated guiding texture coordinate* $\mathbf{q}_G$*, which is the same for the cluster. This way, texture coordinates will stay coherent within the cluster.*

seeded such that the associated clouds are compact, sized above turbulence length scale, and cover all flow paths.

While more sophisticated models for texture advection, e.g., [Yu et al., 2009], or a dynamic re-assignment of guiding particles could be used, we find that the described approach works well in practice. In our example scenes, we seed between 1 and 10 guiding particles per timestep, randomly distributed across the seeding area. We found this to be sufficient to prevent visual artifacts due to coherence loss. For more complex flows, one can revert to global coordinate resetting as described in § 3.5.2 using two sets of texture coordinates, which provides coherence even under strong deformations. This may however introduce some additional diffusion, due to the interpolation between two texture lookups.

The complete simulation loop is specified in the pseudo-code in Fig. 4.5.

4.4 Results and Discussion

In the following, we will discuss several simulations to highlight the features of our model and differences to previous work.

Comparison with reference simulation In order to evaluate the realism of our model, we simulate the flow in the wake of a car (Fig. 4.7). The simulation uses 1M particles, and a base solver resolution of $32 \times 8 \times 32$. We compare our model to a $256 \times 64 \times 256$ high-resolution reference solver. While the exact

Figure 4.7: *The wake behind a car is simulated with 1M particles. Our method (top) and the reference high-resolution solver (bottom) show similar small-scale details.*

form of the turbulence is different between our method and the reference solver, we observe that both show a similar level of small-scale detail.

Energy model We demonstrate the ability of our model to handle obstacle-induced turbulence by simulating a flow over a ramp shown in Fig. 4.11. This setup uses a resolution of the base solver of $64 \times 16 \times 16$ grid cells. When using low grid resolutions such as this, flow instabilities induced by obstacles are dampened out, and no turbulence is induced. This effect can be seen in the top image of Fig. 4.11. In this example, turbulence should develop to the left of the ramp as the flow travels from right to left. Our method tracks causality in the production of turbulence, resulting in a correct swirling motion perpendicular to the edge of the step, purely behind the sharp edge. Turbulence synthesis methods such as Kim et al. [2008b] that amplify or derive turbulent energy directly from the computed velocity field do not track the causality in the production of turbulence. In this case, Wavelet

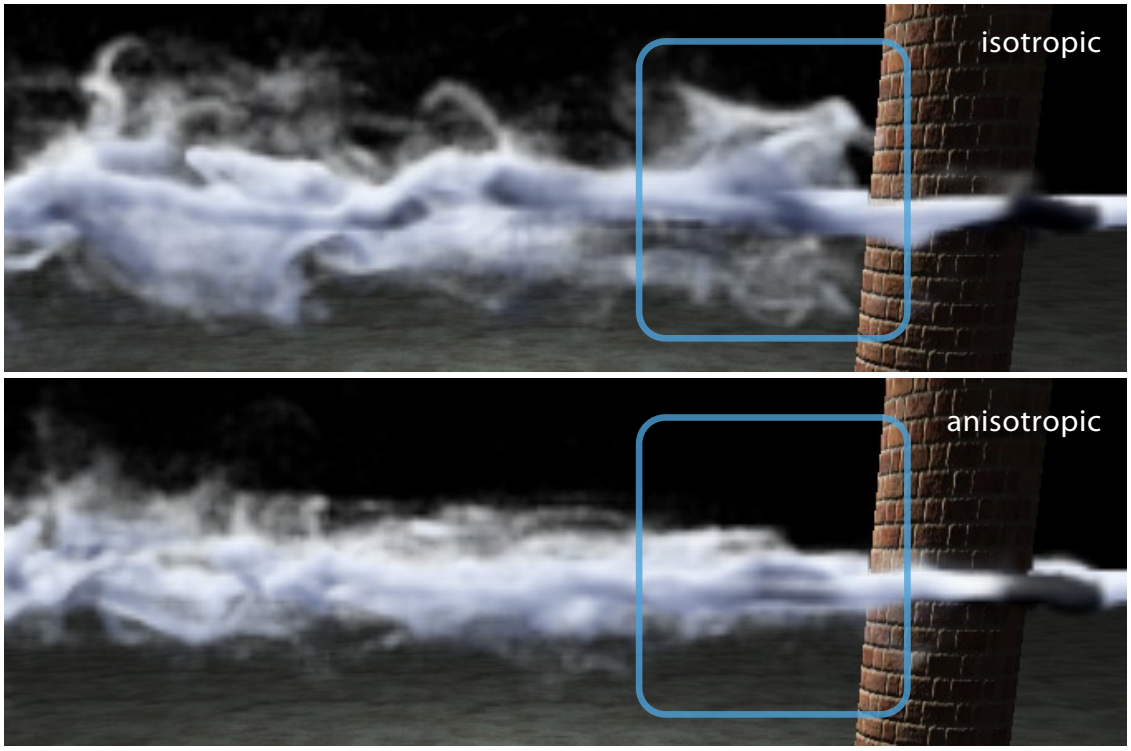


Figure 4.8: *In this example, a thin sheet of smoke flows around a cylinder. Here, the side view is depicted. Using only the isotropic turbulence model (top), the induced turbulence disturbs the flow, as can be seen by the unrealistic spikes left of the cylinder. Using our anisotropy extensions (bottom), the turbulence integrates into the overall flow, and a smooth transition to full isotropic turbulence can be observed.*

turbulence incorrectly produces turbulence in the laminar region right of the edge.

In a more complex example shown in Fig. 4.3, we simulate a train accelerating and braking. Here, the source of turbulence is not induced by obstacles, as in the ramp example, but is due to the pulsed emission of smoke from the chimney. This is also inherently handled by the production term of our energy model. Also, correct adaptation of turbulence intensity to the train's velocity can be observed.

Anisotropy The effect of anisotropic turbulence is demonstrated in a simulation of a strongly turbulent flow past a cylinder. We seed a thin horizontal sheet of smoke to the right, visualizing only a slice of the 3D problem. The side-view of the simulation, with anisotropy handling disabled (top) and enabled (bottom) is shown in Fig. 4.8. If anisotropy is not handled, isotropic

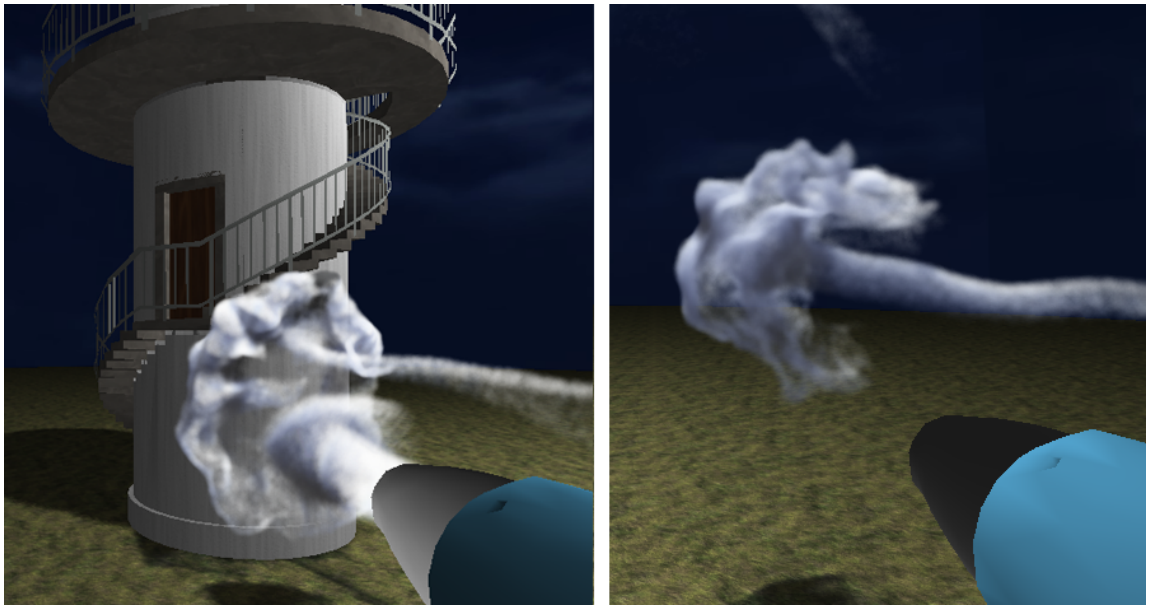


Figure 4.9: *In this game-like scenario, we show that our approach works well for real-time applications. The algorithm is easily integrated in the rendering pipeline, and we are able to achieve frame-rates of 18 frames per second for the complete game, including simulation and rendering.*

turbulence is injected immediately downstream of the obstacle. This leads to strong disturbances normal to the plane of motion, as can be seen in the top image of Fig. 4.8. Our model predicts a zone of high anisotropy behind the cylinder. Here, the turbulence is expected to be confined within the smoke sheet, therefore integrating with the large scale Karman vortices, before becoming more and more isotropic, towards the left side of the lower image.

Scalability To demonstrate the scalability of our model, we simulate a smoke wake behind a car with varying particle numbers, while keeping the grid resolution fixed at $32 \times 8 \times 32$. As can be seen in Fig. 4.1, the large scale flow remains consistent in all cases, while the amount detail is controlled by the number of particles. As it is sufficient to use a very low grid resolution for the Eulerian solver in all examples, the performance scales approximately linearly in the number of particles. With one million particles, our model achieves 15 frame per second on average (including rendering). Increasing the number of particles to four millions, we still achieve 4.7 frames per second. The exact numbers can be found in Table 4.1. This means our model is able to compute accurate turbulence dynamics efficiently. GPU-based methods relying on grids are strongly limited in detail due to the available memory,

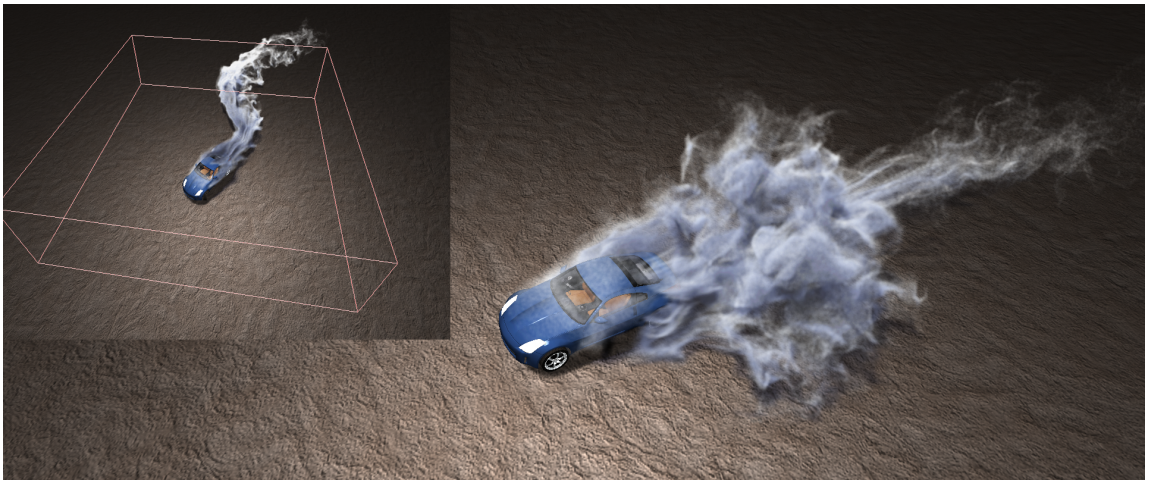


Figure 4.10: *The smoke in this turbulent wake is represented using a particle system. As we synthesize turbulence directly on the particle system, the particles may leave the simulation domain, shown as a red box in this image.*

the Eulerian solver of our implementation, e.g., is limited to a 128^3 resolution using the same hardware. Using our particle based approach we are, on the other hand, able to achieve very detailed motion in an efficient manner.

An example of our method in an interactive game-like setting can be seen in Fig. 4.9. The user controls a smoke emitting gun, demonstrating free stream turbulence as well as turbulence induced by obstacle interaction. Our method also opens up the possibility to compute and synthesize turbulence outside the grid-based solver. If no underlying grid is present, zero turbulence production and the last encountered large-scale velocity are taken as an input for the calculation. This allows a smoke volume to leave the domain of the Eulerian simulation, while still exhibiting turbulent motion, as shown in Fig. 4.10. This is very useful for interactive applications where the spatial limits of the domain should be hidden from the user.

For all of our examples, we vary only the α and L_{inflow} parameters. Recall that α controls the overall amount of turbulence and L_{inflow} controls turbulence at an inlet. Varying only these parameters allows for artistic control while retaining visual realism.

4.5 Conclusions

In this chapter, a novel scalable algorithm for simulating anisotropic turbulence was introduced. By separating the system into a grid-based solver and a decoupled particle system without particle-particle interactions, our

Setup	**Grid res.**	**#part**	α	L_{in}	**Base** [ms]	**Part.** [ms]	**Total** [fps]
Car (Fig. 4.1)	$32 \times 8 \times 32$	250k	2.5	0.04	20	7.6	34
Car	$32 \times 8 \times 32$	1M	2.5	0.04	19	27	15
Car	$32 \times 8 \times 32$	4M	2.5	0.04	20	92	4.9
Car (no turb.)	$32 \times 8 \times 32$	1M	–	–	19	6.4	20
Ramp (Fig. 4.11)	$64 \times 16 \times 16$	1M	2.6	0.08	19	23	17
Aniso. (Fig. 4.8)	$64 \times 16 \times 16$	1M	15.0	0.1	19	27	15
Iso. (Fig. 4.8)	$64 \times 16 \times 16$	1M	15.0	0.1	14	27	16
Smoke gun (Fig. 4.9)	$48 \times 48 \times 48$	1M	4.2	0.02	25	18	18
Train (Fig. 4.3)	$64 \times 32 \times 16$	6M	3.0	0.05	44	161	3.7

Table 4.1: *Performance numbers for our simulation runs. Timings are given per frame.* Base *refers to the grid-based solver, while* Part. *represents turbulence computation, synthesis and particle system update. The total framerate includes both simulation and online rendering. All simulations were run on a NVidia GTX 480 graphics card on a workstation with an Intel Core i7 CPU and 8GB of RAM.*

method is highly efficient on parallel systems. The algorithm is driven by an anisotropic energy transport mechanism, and handles both free stream turbulence production and turbulence induced by walls. Turbulence is synthesized directly on the rendered particles, which allows the simulation to handle the full detail that will later on be displayed, while not wasting any processor cycles for regions that are not visible. This way, we achieve frame rates of more than 15 frames per second even for detailed simulations with millions of particles.

On the other hand, our approach is the restriction to single phase fluid simulations. Also, the algorithm does not perform well for large, non-turbulent smoke volumes, which can have unnecessarily large numbers of particles inside the volume that hardly move. Both of these points are interesting venues for future research. It would be highly interesting to extend our model to free surface flows for liquids, and use an adaptive particle representation to handle larger smoke volumes more efficiently.

In addition, we plan to extend our framework to automatically adapt the level-of-detail for large interactive scenes, as the modularity of our approach makes it highly suitable to combine different simulation approaches. This will allow us to smoothly transition from a simple static flow field, to a Eulerian

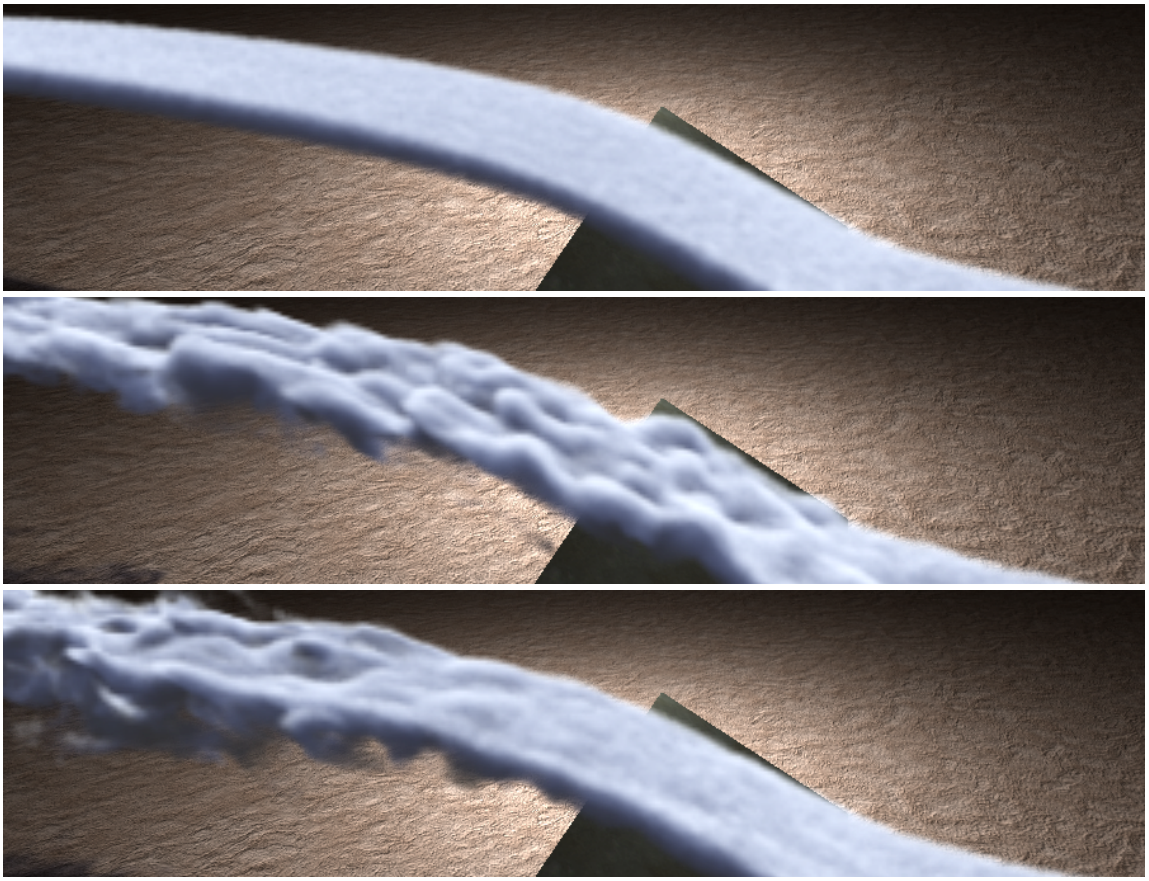


Figure 4.11: *A flow over a ramp is simulated. The low-resolution solver (top) does not represent the flow instability after the edge of the ramp. Therefore methods like Wavelet Turbulence (middle) that depend on the solver for energy calculations also fail to catch the correct turbulence seeding region. Our model (bottom) is able to predict turbulence production due to a full energy transport model.*

fluid simulation, while finally adding detail with our anisotropic Lagrangian turbulence model.

Another limitation is that our algorithm can exhibit artifacts when the underlying simulation is not able to resolve all features of a flow, e.g., in the presence of very thin objects. As our underlying coarse grid solver [Cohen et al., 2010] can only handle first-order accurate boundary conditions, stair-step artifacts may appear around solid curved obstacles. It would therefore be interesting to pair our method with a more accurate real-time solver, and to extend our particle based simulation to handle sub-grid geometric detail.

Finally, this approach shares the limitations of all statistical turbulence synthesis methods, in that turbulence transition is not well-represented. The

breakdown of coherent structures to turbulence is not easily modeled in a statistical manner. As our extension for anisotropy guarantees that the overall shape of the turbulence distribution behaves accordingly, it is able to produce convincing results for fast-developing turbulence. The statistical approach will fail, however, for slow turbulence breakdown, which is visible e.g. in the interface of dense buoyant smoke plumes, or in instable flows.

The methods developed in the two following chapters will address this inherent issue. By directly modeling the turbulence transition process using vortex methods, they allow to represent a wide range of turbulent effects that have not been accessible to turbulence methods.

CHAPTER 5

Modeling Obstacle-Induced Turbulence

Many interesting forms of turbulent flow originate from the complex interaction of flows with obstacles. At the wall of these obstacles, a very thin boundary layer forms, which may separate from the wall, become instable and form free turbulence. As these processes occur on a lengthscale below the resolution of most simulations, the turbulence generation process is often misrepresented or simply omitted. Even turbulence reference scenarios from CFD, such as driven cavities or passive grids are only represented correctly using very expensive high-resolution simulations.

In this chapter, we will develop a method to model this process and predict turbulence generation by flow obstacles. Instead of synthesizing turbulence using a frequency-matched curl noise texture as in § 4, we will directly represent the turbulence eddies using vortex particles, which enables coherent anisotropic structures in e.g. turbulence transition. Together with prediction, this will allow us to represent a wide range of complex turbulence effects, which can not be achieved using the general-purpose turbulence model presented in the previous chapter. As we are able to precompute the formation process for a given obstacle geometry, we can even correctly predict turbulence generation from obstacles thinner than grid resolution (Fig. 5.1).

While turbulence detail enhancement using vortex particles has been previously studied in Graphics [Selle et al., 2005], these methods deal with the

Figure 5.1: *Our algorithm allows us to precompute detailed boundary layer data and efficiently reuse it for new simulations. We are able to generate turbulent vortices taking into account the relative velocity of an obstacle in the flow. Here, we apply our algorithm to a very thin object that is barely represented on the simulation grid.*

preservation of vortices manually injected in the flow. We use an enhanced version of the vortex particle approach and combine it with turbulence methods to guarantee that particles are accurately seeded and the energy dynamic adheres to what is predicted by CFD turbulence theory. Moving the computations for the generation of turbulence into a preprocessing step allows us to quickly set up new simulations around a given object. The contributions of our method are:

- A new method to accurately track and precompute boundary layer vorticity using an *artificial boundary layer* whose accuracy is independent of a final simulation resolution.
- An algorithm to process the precomputed data in a dynamic simulation to spawn vortices according to the current flow.
- A vortex particle method that models vortex interactions and adheres to turbulent energy transport theory.

5.1 Overview

The algorithm presented consists of a precomputation step for the scene geometry, and a simulation step.

The precomputation step captures the characteristics of the boundary layer around the object, and stores it for different sets of flow directions. This allows us to purely resolve the geometry of the object with the precomputation, instead of having to fully resolve the actual flow velocity in the often very thin boundary layer. For this precomputation, we assume that an object can be characterized by a relative translational and rotational velocity, allowing for simulations of rigid body motion or static flows of arbitrary direction.

The main simulation method consists of a standard, grid-based fluid solver, e.g. according to Stam [1999], augmented with a turbulence representation. The precomputed boundary layer data is used to efficiently calculate where vortices are created around the object in a separate simulation. We compute the evolution of boundary layer around the object, and estimate regions where this field becomes unstable to form actual turbulent vortices. The turbulent vortices are represented using an improved variant of vortex particles [Selle et al., 2005], which induce rotation in the flow around the particle position. While the vortex particles are created based on boundary layer vorticity, their dynamics is based on the vorticity equation. In an additional step, we remesh the particles and adjust the particle kernel to ensure a correct turbulent energy distribution. The resulting vorticity is finally reconstructed onto a grid with higher resolution than the base simulation.

The key point of this chapter is the turbulence estimation and vortex particle seeding mechanism. In § 5.2, we will develop a theory for this. In § 5.3, the dynamics of the vortex particles are described, and the coupling of the vortex particles to the flow field is explained. The actual simulation loop and implementation details are discussed in § 5.4. The simulation loop is also visualized in Fig. 5.2. Finally, we will evaluate the method and present results in § 5.5.

5.2 Wall-Induced Turbulence

While turbulence in flows is generated by various processes, a very common and visually important one is turbulence generation at the flow boundaries. Therefore, our algorithms explicitly model this important process. In this section, our turbulence estimation method is introduced. It bases on turbulence

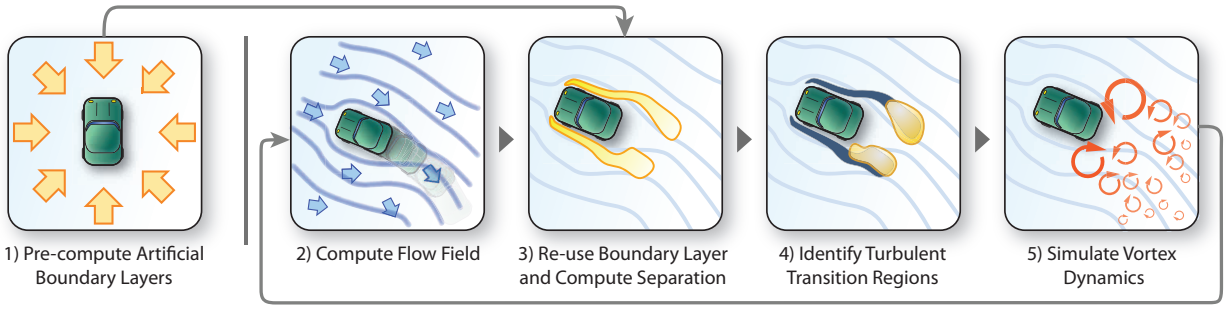


Figure 5.2: *An overview of different steps of our algorithm. After precomputing the artificial boundary layer (1), we run or not a new simulation (2) and apply the confined vorticity from the precomputation (3). Regions transitioning into turbulence are identified with an approximation of the Reynolds stress (4). This results in the creation of vortex particles. Their dynamics are computed in an additional step (5).*

modeling and wall flow theory, which is introduced in § 3.3. A more detailed account can be found in the book by Pope [2000].

5.2.1 Generation of turbulence

In wall-bounded flows, wall friction enforces a tangential flow velocity of zero at the wall. This leads to the formation of a thin layer with reduced flow speed, called the boundary layer. Fig. 5.3 shows a velocity profile in the boundary layer. It has been shown that this profile is equivalent for all wall-bound flows when using normalized units. This *universal law of the wall* was stated by van Driest in [1956].

The gradient of tangential flow velocity in the boundary layer leads to the creation of a thin sheet of vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. For planar walls, this vorticity remains mostly confined to the boundary layer, and we will thus refer to it as *confined vorticity*. At regions of high flow instability however, vorticity may be ejected from the boundary layer and enter the flow as turbulence. This happens e.g. at sharp edges, where the boundary layer is separated from the wall, and likely to become unstable, or when other turbulent structures disturb the boundary layer. This process of turbulence formation is referred to as roll-up, and is the predominant mechanism of wall-induced turbulence generation [Jiménez and Orland, 1993]. There is no theory quantitatively describing the boundary layer roll-up process. We will therefore model this process in a statistical sense, as explained below.

Turbulence modeling We base our approach on CFD turbulence modeling techniques. These techniques model statistical properties of turbulence based on the ensemble-average of the flow field $\mathbf{u}$. For a quasi-static flow, ergodicy permits us to use the time-averaged flow field, i.e. the flow field with all fluctuating turbulent structures averaged out, instead of the more complicated ensemble averaging.

One of the most important quantities that can be modeled in such a way is the Reynolds stress tensor. It governs the transfer of energy from the bulk flow to turbulent structures, and thus the generation of turbulence. This fact is commonly used for RANS or LES simulations, and we will make use of it for our method as well. Next, we will describe how we model the boundary layer.

Boundary layer modeling In order to accurately model wall-induced turbulence formation, we need to track the confined vorticity, simulate the boundary layer separation and finally identify the transition points to turbulence.

As the boundary layer attached to an obstacle is very thin (smaller than simulation grid resolution in most cases), it is difficult to directly measure the confined vorticity. Instead, we leverage the universal law of the wall, and note that the confined vorticity only depends on the velocity scale and materials constants. For each point in the wall-attached boundary layer we therefore determine the confined vorticity as

$$\boldsymbol{\omega}_{ABL} = \beta(\mathbf{U}_s \times \mathbf{n}) \,. \tag{5.1}$$

The velocity scale $\mathbf{U}_s$ is the tangential component of the averaged flow velocity just outside the boundary layer. The constant β accounts for the two material constants, skin friction coefficient and the fluid viscosity. In our model, β is a user-defined parameter. We call the resulting field $\boldsymbol{\omega}_{ABL}$ the *artificial boundary layer*.

On the other hand, boundary layer separation is an advective transport process. If the wall-attached part of the artificial boundary layer is known, then the separation plume can be derived by advecting this field with the flow field during the simulation run.

The last missing part is to identify regions where the separated boundary layer becomes unstable, and the confined vorticity $\boldsymbol{\omega}_{ABL}$ transitions to free turbulence. The anisotropic part of the Reynolds tensor a_{ij}, which is responsible for the production of turbulence, is a good indicator for such transition

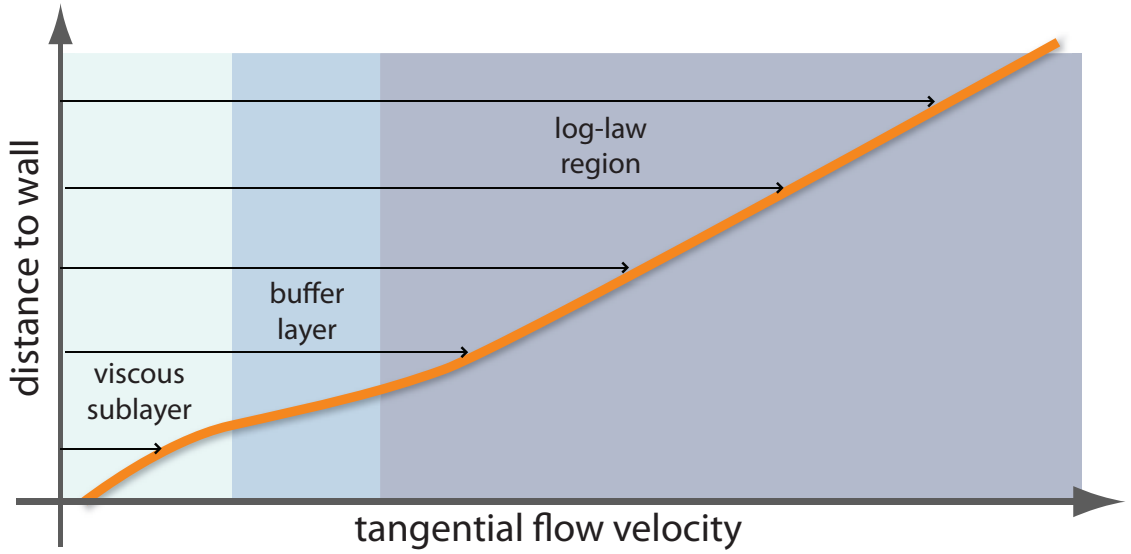


Figure 5.3: *The mean velocity profile near a wall (in normalized units) has the form shown above. This has been confirmed in numerous experiments, and was formulated as a universal law by van Driest.*

regions. We therefore define a transition probability density p_T, which is used to seed turbulence,

$$p_T = c_P \Delta t \frac{\|a_{ij}\|}{|\mathbf{U_0}|^2} \tag{5.2}$$

such that regions with high Reynolds stresses are likely transition regions. Here, $\|\cdot\|$ denotes the Euclidean matrix norm. Reynolds stresses are normalized to a uniform scale by the inflow velocity $\mathbf{U}_0$, and c_P is a parameter to control the seeding granularity. If using varying time-steps, p_T has to be multiplied by Δt to ensure consistent behavior. In the following, we will describe how to compute the Reynolds stress tensor based on stresses in the averaged flow field.

Reynolds models The anisotropic component a_{ij} of the Reynolds stress tensor R_{ij} can be expressed using the turbulent viscosity hypothesis

$$a_{ij} = -2\nu_T S_{ij} \ , \tag{5.3}$$

where ν_T is the turbulent viscosity and S_{ij} denotes the strain tensor. The turbulent viscosity can be expressed in terms of a *mixing length* l_m. We chose the model of Baldwin [1978] for modeling the turbulent viscosity which states

$$\nu_T \approx {l_m}^2 \|\Omega_{ij}\| \ . \tag{5.4}$$

with the rotation tensor Ω_{ij}. While the mixing length is not known for the general case, we will only apply this model in the near-wall region, where l_m is known to be linear in wall distance. Using these standard methods, it is possible to predict the generation of turbulence using only the non-turbulent mean flow velocities.

However, the presented Reynolds stress model requires a high grid resolution around the boundaries to capture the thin boundary layer accurately. In a typical fluid simulation in graphics, the boundary layer thickness is often smaller than a grid cell. Consequently, the discrete S and Ω operators will fail to capture the desired effect, or even cause instabilities due to highly discontinuous gradients, as also mentioned by, e.g. Narain [2008].

We therefore propose two changes to this model. First, we know that in regimes close to a wall, the norm of the rotation tensor equals the norm of the confined boundary layer vorticity, $\|\Omega_{ij}\| = |\boldsymbol{\omega}_{ABL}|$. Also, we assume that $\|S_{ij}\| \approx \|\Omega_{ij}\|$. This is a good approximation if the velocity gradient is dominated by the component normal to the wall [Pope, 2000], which, except for sharp corners, is usually the case in the near-wall region. With these assumptions, we can rewrite the Reynolds stress without the problematic discrete stress and rotation tensors as

$$\|a_{ij}\| \approx 2{l_m}^2 |\boldsymbol{\omega}_{ABL}|^2. \tag{5.5}$$

Combined with Eq. (5.2) this leads to the final equation for the transition probability.

$$p_T = 2\, c_P\, \Delta t\, {l_m}^2\, \frac{|\boldsymbol{\omega}_{ABL}|^2}{|\mathbf{U}_0|^2}\,. \tag{5.6}$$

The seeding process for vortex particles, based on p_T, is explained in § 5.3.2.

5.2.2 Precomputing the Artificial Boundary Layer

The artificial boundary layer together with Eq. (5.6) can be used to seed turbulence, in the form of vortex particles, in the appropriate places of the flow. However, the expression for the wall-attached $\boldsymbol{\omega}_{ABL}$ depends on the averaged flow field $\mathbf{U}$, which is not accessible during the simulation. It is not possible to use the instantaneous flow field of the simulation, as the emerging turbulence would lead to feedback loops. However, we can precompute $\boldsymbol{\omega}_{ABL}$ for quasi-static scenes or scenes with rigidly moving objects. This has the additional advantage that we can choose simulation resolution and precomputation resolution independently, allowing us to precompute fine boundary geometries, while running the simulation on a coarse grid.

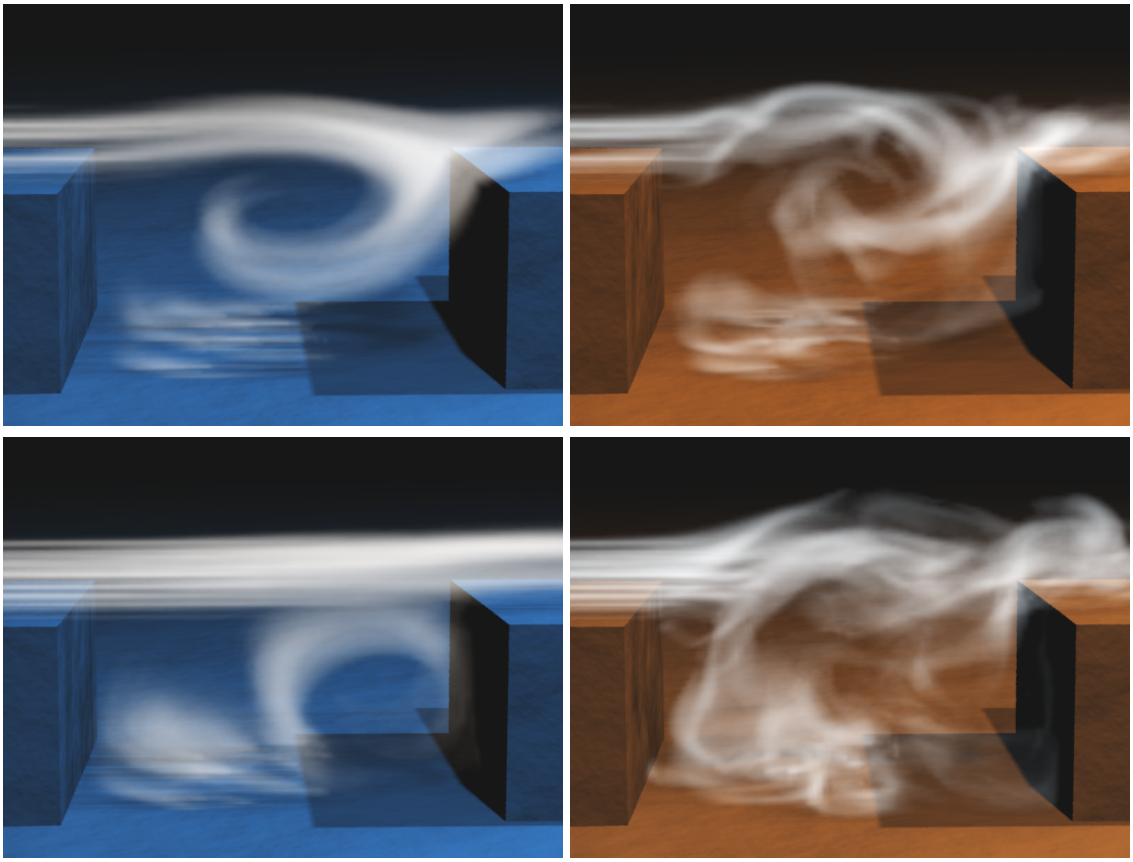


Figure 5.4: *The top picture show a basic simulation of a fluid flowing left to right over a cavity. This flow produces a big vortex in the cavity, but is unable to capture any generation of turbulence from the walls. With our method (pictures on the right) we are able to identify the confined vorticity shedding off the two edges of the cavity, and introduce corresponding vortex particles to represent the turbulent structures forming in the flow.*

Precomputation is done by running a standard fluid solver, and time-averaging the flow field. At all obstacle boundary voxels, Eq. (5.1) is evaluated, and $\boldsymbol{\omega}_{ABL}$ is stored in a suitable data structure (see pseudo-code Fig. 5.5). More details on the implementation of the precomputation step, and how the precomputed data is used in the simulation will be given in § 5.4. In the next section, we will explain how to compute the dynamics of our turbulence representation.

```
1: Perform standard grid-based simulation
2: Obtain time-averaged flow field U
3: for each voxel x on the obstacle boundary do
4:    // Get voxel outside the boundary layer
5:    x_e ← x + l n
6:    ω_PRE ← β(U(x_e) × n)
7:    Store (x, ω_PRE) in a point set
8: end for
```

Figure 5.5: *Pseudo-code for precomputing the Artificial Boundary Layer.* $\mathbf{n}$ *denotes the surface normal and* l *is the boundary layer thickness.* l *is chosen to be the distance from the wall at which the velocity gradient approaches zero, usually 1-2 grid cells.*

5.3 Turbulence Synthesis

We chose to synthesize turbulence using vortex particles. In contrast to curl noise-texture based turbulence methods, vortex method directly represent the turbulent structures, and therefore automatically preserve coherence. Transition processes can directly be modeled, as they allow more degrees of freedom in anisotropy. Sparse particles allow us to focus on sampling the regions where turbulence is actually generated. Narain et al. [2008] use particles with curl noise textures as a turbulence representation. However, the blending of noise textures creates diffusion, and this approach only supports isotropic turbulence. As we want to model highly anisotropic generation processes, and extend our model into the model-dependent range, where no uniform direction and energy distribution can be assumed, we use an enhanced variant of the vortex particle method by Selle [2005] instead. In contrast to the original paper, we also model energy transfer and make use of an improved synthesis step.

5.3.1 Vortex particle dynamics

Turbulence dynamics can be seen from two points of view: The vorticity differential equation describes the direct evolution of the vorticity field, while the energy transport equation describes its statistical behavior. Both consist of terms for advection, generation, dissipation and scale transfer, but have different advantages for a Lagrangian representation. While the vorticity equation is well suited for describing dynamics, the injection and dissipation of energy via particle creation and dissipation is easier in an energy formu-

lation. We will use a combination of both representations to leverage the strengths of both models.

Motion equation The motion of vortex particles is described by the vorticity equation Eq. (3.4). However, we will use the vortex particles not as a full representation of the velocity field, but use it in combination with a grid-based Navier-Stokes solver. The velocity field $\mathbf{u}$ therefore consists of two parts, the flow field of the grid solver $\bar{\mathbf{U}}$ and the detail velocity induced by the vortex particles. We leave external forces and baroclinity to the underlying solver, which will affect the vortex particles via its velocity field. With this, the evolution for the vortex particles becomes

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega} \,. \tag{5.7}$$

The left side of the equation is handled by advecting the particles in the final high-res flow field augmented with turbulence. The first term on the right-hand side is the vortex stretching term. It is computed by trilinear interpolation of the discrete gradient of the velocity grid, and is used to adjust the particles' vorticity magnitude by $\Delta t(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$. This term is problematic as it might introduce exponential accumulation of vorticity magnitude in a particle. Therefore, the particle is rescaled after the update to preserve the magnitude, effectively only spinning the particle, but not altering its strength. The strength, effectively a measure of energy gain and loss, is handled by the energy dynamics, which is explained in the next section. Similarly, the viscous diffusion term (the second term on the right hand side of Eq. (5.7)), will be handled by energy dynamics, as it is not easily represented on a sparse particle system.

This gives us a reduced formulation of Eq. (5.7) which conserves vorticity as well as energy. It is therefore orthogonal to the energy transfer equations, the computation of which we will describe next.

Energy dynamics To model the transfer of energy, we will model energy transport in the sense of a turbulence model § 3.3. As the turbulent energy in our system is represented using vorticity, not kinetic energy k, it is not practical to directly solve the energy transport equation of a turbulence model. Instead, we express the individual energy transport terms to our vorticity model and apply it directly on our particle system. In its most general formulation, the turbulent energy transport equation states

$$\frac{\partial k}{\partial t} + (\mathbf{u} \cdot \nabla)k = -\nabla \cdot \mathbf{T} + \mathcal{P} - \epsilon \quad . \tag{5.8}$$

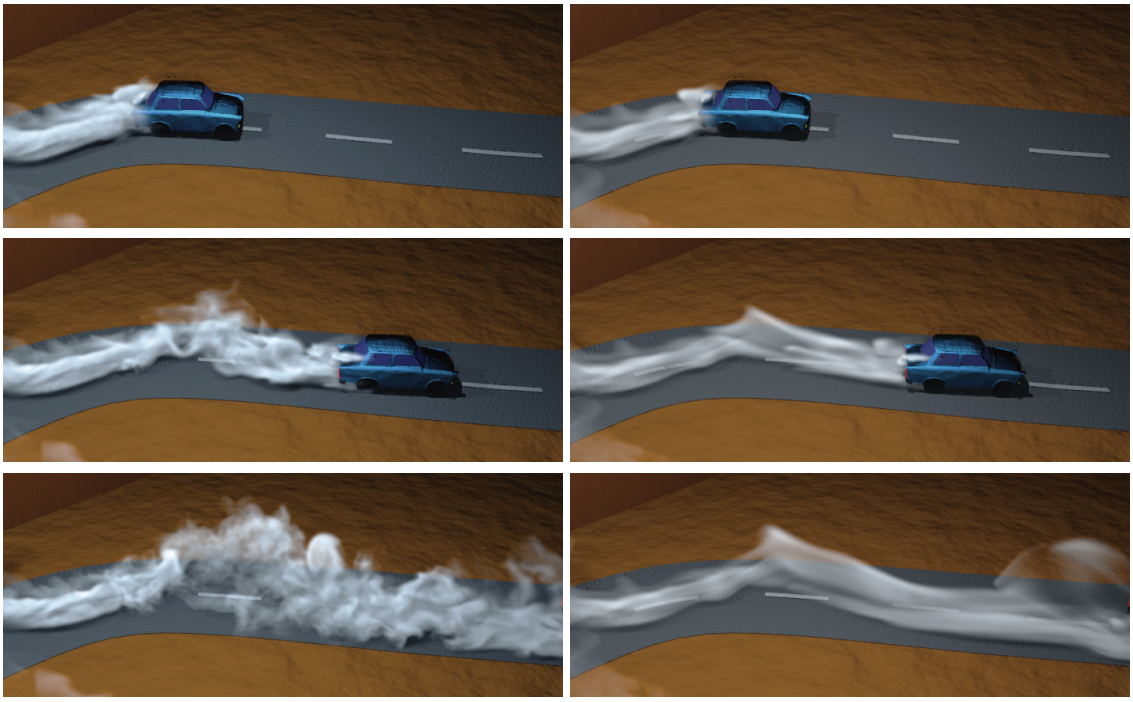


Figure 5.6: *Example of a moving object inducing turbulence in its wake. On the left, the base simulation is shown. In the images on the right, this simulation has been augmented with vortex particles using our method. As the car accelerates (middle and lower picture), more turbulence is expected – this behavior is correctly predicted by our model.*

The left hand side again is represented by advection of the particles. The right-hand side consists of production $\mathcal{P}$, dissipation ϵ and the energy transfer term $\nabla \cdot \mathbf{T}$, which is approximated using the gradient diffusion hypothesis in classical turbulence models. Outside the inertial subrange, this quantity is however hard to model. Its behavior will therefore be based on the length scale, as explained below.

For the production term, we can use the information from our artificial boundary layer (see § 5.2.2). The dissipation ϵ occurs at wavenumbers that are usually well below the resolved grid resolutions. Dissipation is therefore implemented by removing particles whose radii are too small to be represented on the grid. We use a threshold of $2\Delta x$ for our simulations. Finally, for handling the remaining energy transfer term for $\mathbf{T}$ of Eq. (5.8), we distinguish the following two cases:

1. *The particle is in the inertial subrange.* We represent the energy cascade by decaying a particle with wavenumber κ_a into n particles of smaller wavenumber κ_b. We typically use n=2 in our simulations. From Kolmogorov's law,

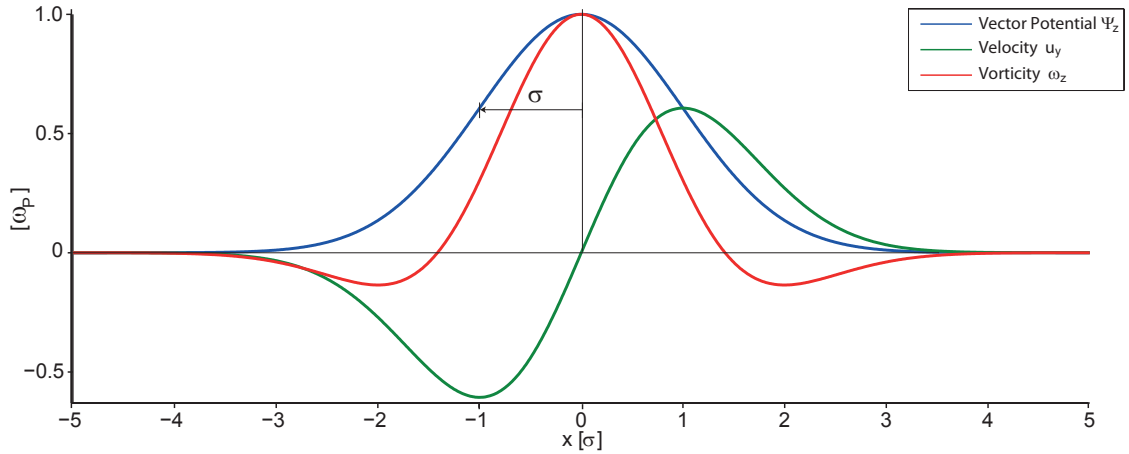


Figure 5.7: *Vector potential, velocity and vorticity of the vortex particle kernel are shown along a x-axis slice.*

we can derive a timescale of decay as $\Delta t = C(\kappa_a^{-\frac{2}{3}} - \kappa_b^{-\frac{2}{3}})$, where C denotes a parameter that depends on the rate of dissipation ϵ. In practice we can use a value normalized by the averaged flow $\mathbf{U}$ here. We also know that for the turbulent energies $k_a/k_b = n\,(\kappa_a/\kappa_b)^{-\frac{5}{3}}$ holds, which is used to derive the vorticity magnitude of the new particles. For practical reasons, we also add a small position and angle displacement to the new vortex particles, as they would otherwise lump together.

2. *The particle is in the model-dependent range.* As transfer cannot be easily described in this regime, a heuristic is used. Typically, small vortices with aligned direction tend to form larger vortices in this range. Therefore, we merge vortex particles in the model-dependent range with a distance of less than the particle radius to a single larger vortex particle. Here, the vortex magnitude is chosen so that total the energy is conserved as $k_{new} = k_1 + k_2$. As very small and strong vortex particles might induce stability problems, we also conserve the energy density, i.e. $\frac{k_{new}}{V_{new}} = \frac{k_1}{V_1} + \frac{k_2}{V_2}$. This specifies the radius and strength of the merged particle. The new direction is obtained by a weighted average, with the respective energies as a weight.

5.3.2 Vorticity Synthesis

To synthesize turbulence from the vortex particles, we need to obtain a detail velocity field from the particles system. Each vortex particle has a vorticity vector $\boldsymbol{\omega}_P$, encoding magnitude and rotation axis, and a kernel over which this value is applied. The detail field can be obtained by integrating this vorticity kernel and mapping it on a higher-resolution grid.

Kernel For the direct regulation of vorticity, a kernel with the following properties is desired: at the vortex particle center, vorticity should be equal to $\boldsymbol{\omega}_P$. Also, the resulting vector field should mainly contain rotation around $\boldsymbol{\omega}_P$, and smoothly fade out with the particle radius without causing discontinuities. And lastly, the associated velocity field, and its integral, which is needed for e.g. energy calculation, should be a simple analytic form. We chose a Gaussian peak with standard deviation of σ in the vector potential to meet these requirements. In cylindrical coordinates, it is given by

$$\Psi(z,\varphi,\rho) = -|\boldsymbol{\omega}_P|\,\sigma^2 \exp^{\frac{-\rho^2-z^2}{2\sigma^2}} \mathbf{e}_z\,, \tag{5.9}$$

where the vortex axis $\mathbf{e}_z$ is aligned with $\boldsymbol{\omega}_P$. We can then derive the velocity field

$$\mathbf{u} = \nabla \times \Psi = -|\boldsymbol{\omega}_P|\,\rho \exp^{\frac{-\rho^2-z^2}{2\sigma^2}} \mathbf{e}_\varphi\,, \tag{5.10}$$

and the vorticity kernel

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} = -\frac{|\boldsymbol{\omega}_P|}{\sigma^2}\left(\rho z\,\mathbf{e}_\varphi - (\rho^2 - 2\sigma^2)\mathbf{e}_z\right) \exp^{\frac{-\rho^2-z^2}{2\sigma^2}}\,. \tag{5.11}$$

A cut-off radius is used to make the kernel support finite. We use $r = \sqrt{6}\sigma$ at which point the exponential term of the kernel function has fallen to 10^{-3}. The length scale is defined at the kernels' origin, so that its wavenumber is $\kappa = \frac{1}{\sigma}$. For the contained energy $E \propto {\boldsymbol{\omega}_P}^2\,\sigma^5$ holds.

Synthesis To combine the detail field and the velocity field from the underlying solver, these two fields could be simply added, as in the method in § 4. However, we also want to allow vortex sizes above the grid resolution of the underlying solver, which means turbulent detail might overlap with existing vortices from the base solver. Therefore, we need to exclude vorticity already represented in the base solver to avoid duplication. To achieve this, the base grid vorticity is measured, and only the difference is synthesized on the detail field.

The synthesis is a three-step process: First, the vorticity field $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ of the velocity grid is computed by finite differences. Second, all particle vorticity kernels are summed up to obtain a desired vorticity field $\boldsymbol{\omega}_D$. And third, each particle adds its kernel to the velocity field, scaled by a weight w_k, computed as:

$$w_k = \frac{\sum_{kernel}(\boldsymbol{\omega}_D - \boldsymbol{\omega}) \cdot \bar{\boldsymbol{\omega}}_P}{\sum_{kernel} \boldsymbol{\omega}_D \cdot \bar{\boldsymbol{\omega}}_P}\,, \tag{5.12}$$

where $\bar{\boldsymbol{\omega}}_P$ is the particles' normalized rotation axis. The dot product with $\bar{\boldsymbol{\omega}}_P$ ensures that only the vortex particles' direction is considered, and the kernel

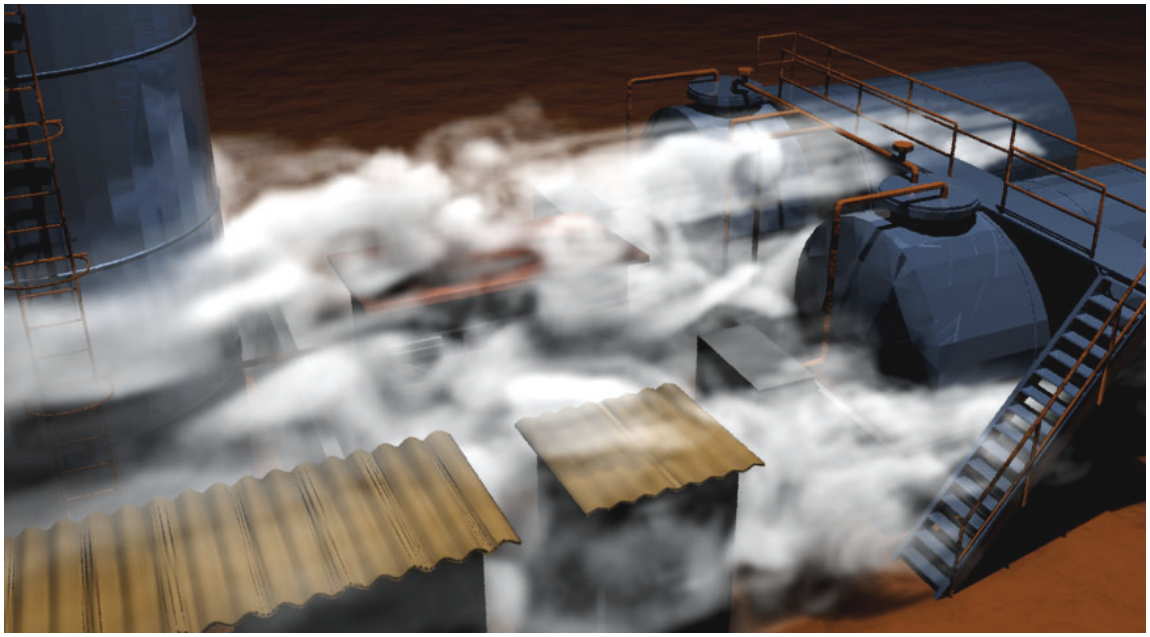


Figure 5.8: *In this example a static flow field is used to generate complex turbulence around an object with our method.*

is normalized by the sum of desired vorticity. We achieve an exact regulation of the vorticity sum under the kernel in one timestep by this process.

Vortex particle seeding As explained in 5.2.2, particles will be seeded in regions of high normalized Reynolds stress. Based on the probability $p_T(\mathrm{x})$ from Eq. (5.6), a particle is created at position $\mathbf{x}$.

All confined vorticity $\boldsymbol{\omega}_{ABL}$ within the particle's radius is removed from the artificial boundary layer, and the particle's strength and direction $\boldsymbol{\omega}_P$ are set such that the vorticity integrated over the kernel equals the removed vorticity sum. We choose the particle radius to be as large as possible without touching an object. We allow for radii up to a size r_{max} which is fully resolved by the main simulation (we have used a value of $r_{max} = 6\Delta x$ below).

The constant c_P in Eq. (5.6) controls the granularity of the seeding process. If set to a high value, confined vorticity is turned into free turbulence relatively quickly. This results in a large number of weaker particles near the object, which then merge to large vortices. On the other hand, if c_P is set to a high value, the artificial boundary layer plume can grow, and fewer, stronger particles form. With an appropriately chosen c_P, numerical cost can be kept low while avoiding popping artifacts that may occur if overly large particles are seeded. We use a c_p of $\approx 1 - 4$ in our simulations.

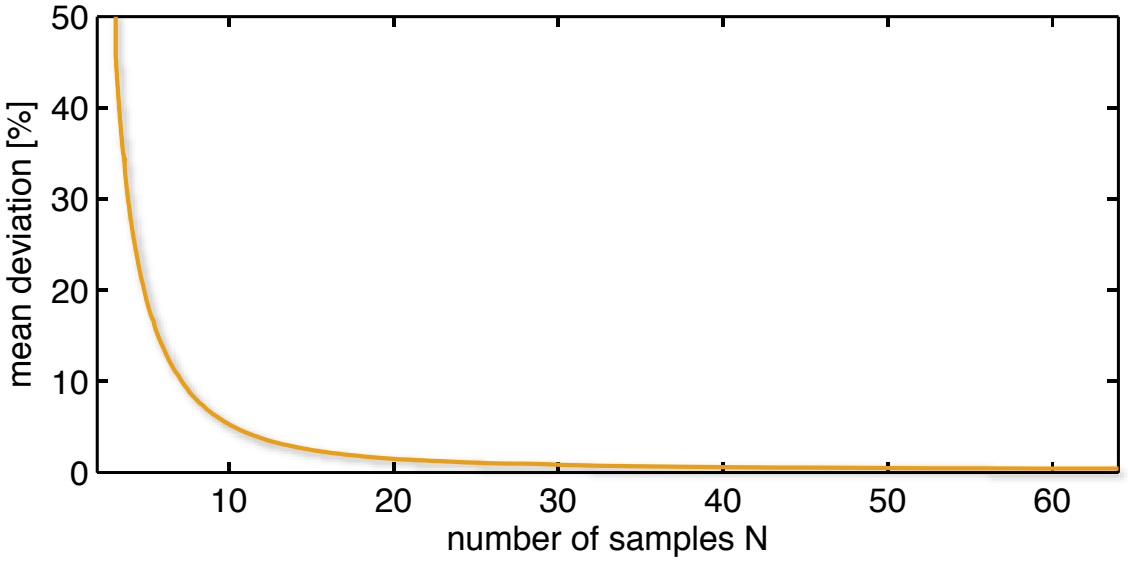


Figure 5.9: *Mean relative error of the artificial boundary layer values for the car model. A reference simulation is compared to a spherical interpolation with N samples for the azimuth.*

5.4 Implementation

In this section, details on our implementation of the precomputation step and the main simulation loop are provided.

5.4.1 Precomputation

For precomputing the artificial boundary layer, it is essential to resolve the mean flow around an object. This can be done using time-averaging over a long period of time with a standard solver, or using a RANS solver. As we are only interested in the velocities around the boundary layer, we have used a standard solver with an artificially increased viscosity in the form of a diffusion step for the velocities. Due to the increased viscous effect, it stabilizes quickly and an average over fewer frames can be used. We have found that using the more complex RANS or longtime-averaging does not pay off visually compared to this more efficient solution. After obtaining the averaged flow field, the boundary layer is calculated according to the pseudo-code (Fig. 5.5) and stored as a point set.

Moving objects To precompute the flow for scenes with moving objects, the boundary layer around each moving objects is precalculated. If the object can

move and rotate freely, or its movement is not known a priori, our algorithm allows us to precompute the whole range of movement directions to later on generate arbitrary simulations of the object in a flow. For this, we split the movement into a translational and a rotational component and precompute artificial boundary layers for each.

To perform the precomputation, the object is placed in the center of a simulation grid. The domain box is chosen large enough not to disturb the flow around the object. For the translational component, we leave the object fixed and use different inflow velocities, defined as boundary conditions on the domain box. As $\boldsymbol{\omega}_{ABL}$ is linear in the velocity magnitude, we only need to sample the velocity direction. In our simulations, we use 10×20 samples in spherical coordinates. For the rotational component, the object is placed in a standing fluid, and we rotate the object with normalized speed around a chosen axis. Again, we use 10×20 samples in spherical coordinates to sample the rotation axis direction.

The simulations stabilize quickly due to the increased viscosity. We have used 50 steps for the examples shown in our video. In the precomputations for the rotational component, this is equivalent to one full rotation. After stabilizing, we average the velocities over another 50 frames. We note that precomputations for each direction can be trivially done in parallel.

Applying the precomputed set At simulation time, we determine the objects linear velocity relative to the scene, and its rotation axis. We then look up the nearest values in the precomputed database. A bilinear spherical interpolation is performed for both the linear velocity direction as well as the rotation axis. The results of the interpolation are scaled by respective magnitude and added. We have performed error measurements for the linear interpolation of the boundary layer values. The corresponding graph can be seen in Fig. 5.9. Our choice of 20 directional samples in the azimuth means we have an interpolation error of 1.6%. As the decomposition into rotational and linear component is only an approximation, we have also measured its error for the car model (Fig. 5.6). In this case the error is 8% on average, and thus small enough not to cause visual artifacts.

It is not necessary to fully resolve the boundary layer during the precomputations, as our model described in § 5.2.1 takes care of this. Instead, one should make sure the resolution of the precomputation is sufficiently fine to resolve all important geometric features of the object. This is eased by the precomputation focusing only on that object, even if it will only occupy a tiny fraction of the final simulation domain. In addition, since the precompu-

Setup	Fig. 5.1	Fig. 5.6	Fig. 5.10	Fig. 5.8
Grid res.	160·70·160	150·40·200	100·25·60	250·80·150
ABL upscaling	2	2	2	1
Frame time [s]	19.5	13.4	10.0	1.3
ABL time [s]	5.5	3.7	0.05	0.7
# particles	∼900	∼700	∼600	∼1000
Vortex gain β	6.2	0.4	3.2	4.0
Precomp. res.	70x150x70	70x70x120	100x25x60	250x80x150
Precomp. [s]	220	112	59	227

Table 5.1: *Detailed statistics for our simulation runs.* ABL upscaling *refers to the up-sampled grid on which vortex particle evaluation and smoke/levelset advection is performed. The precomputation time is given per database parameter.*

tation can be used on many simulations, a high resolution precomputation grid can quickly pay off. In § 5.5 we demonstrate the effectiveness of the precomputation even when an object is extremely thin.

5.4.2 Simulation loop

For the actual simulation, a standard fluid solver and a vortex particle system are coupled. In each simulation step, the artificial boundary layer is updated, new vortex particles are created and vortex particle dynamics are applied. Afterwards, the turbulence forces are added to the flow field, and finally, the remaining steps of the standard fluid simulation are performed. This is repeated each time-step. Pseudo-code for this extended simulation loop can be found in Fig. 5.12

Note that we can also independently choose a higher grid resolution for the evaluation of the vortex particles. This allows us to more accurately evaluate the particle kernels, which is especially useful for small scale vortex details. This high resolution velocity field is down-sampled for the main simulation steps (line 28 in the pseudo-code), and up-sampled for our algorithm before starting with line 1. Performing the algorithm (line 1–25) with a higher resolution enables us to simulate detailed features, e.g., when advecting smoke densities or a free surface level set, while the costly pressure projection operates on a small grid resolution. Typically, we have used a two times higher resolution for the examples below.

5.5 Results and Discussion

In the following section we discuss comparisons of our method to previous work and a reference simulation. In addition, we demonstrate several complex examples of turbulence being generated around moving objects or due to effects such as wind or a flowing river.

Comparisons In Fig. 5.4 the effect of our wall-induced turbulence can be seen for the flow over a cavity with a grid resolution of $120 \times 60 \times 40$. The top image shows a standard, unmodified simulation, while the lower image uses our algorithm to introduce wall-induced turbulence. Both simulations use the same grid resolution, but the unmodified simulation is unable to capture any turbulence being generated from the shearing near the walls. Our simulation exhibits complex vortices due to the vorticity generated at the wall boundaries. To compare our method to approaches for synthetic detail generation, we have simulated the same cavity setup including wavelet turbulence, using the implementation available on the paper's website [Kim et al., 2008b]. This comparison is shown in Fig. 5.11(B). Wavelet turbulence successfully adds small detail to the overall flow, but has difficulties introducing larger vortices to the strong horizontal motion. In contrast, our method introduces persistent larger vortices, while the resulting smoke filaments are successfully broken up by the wavelet turbulence. This shows that our method is suitable for bridging the gap between small synthetic vortices and the vortices resolved by a standard simulation.

We use the setup shown in Fig. 5.11 to compare our method with normal vortex particles [Selle et al., 2005]. The simulations now focus on the left edge of the cavity. The image (A) shows a reference simulation, using a four times increased grid resolution. Note that the flow along the wall to the left is completely straight, while turbulent structures form to the right of the backward facing step. This behavior has been confirmed in various experiments and simulations (e.g. [Le et al., 1997]). The image (C) shows the flow after randomly introducing vortex particles along the walls. Naturally, the vortex particles do not take the overall flow into account, and strongly distort the structure of the flow. Our method, shown in picture (D), is able to recover the vortices being shed off the step, without distorting the flow along the wall to the left. Although we are not able to fully recover the flow of the reference simulation due to the different numerical viscosities, our method is able to qualitatively capture the wall-induced turbulence at a much lower computational cost. On average, the time per frame for the reference simulation was 218 times higher than for the simulation with our algorithm.

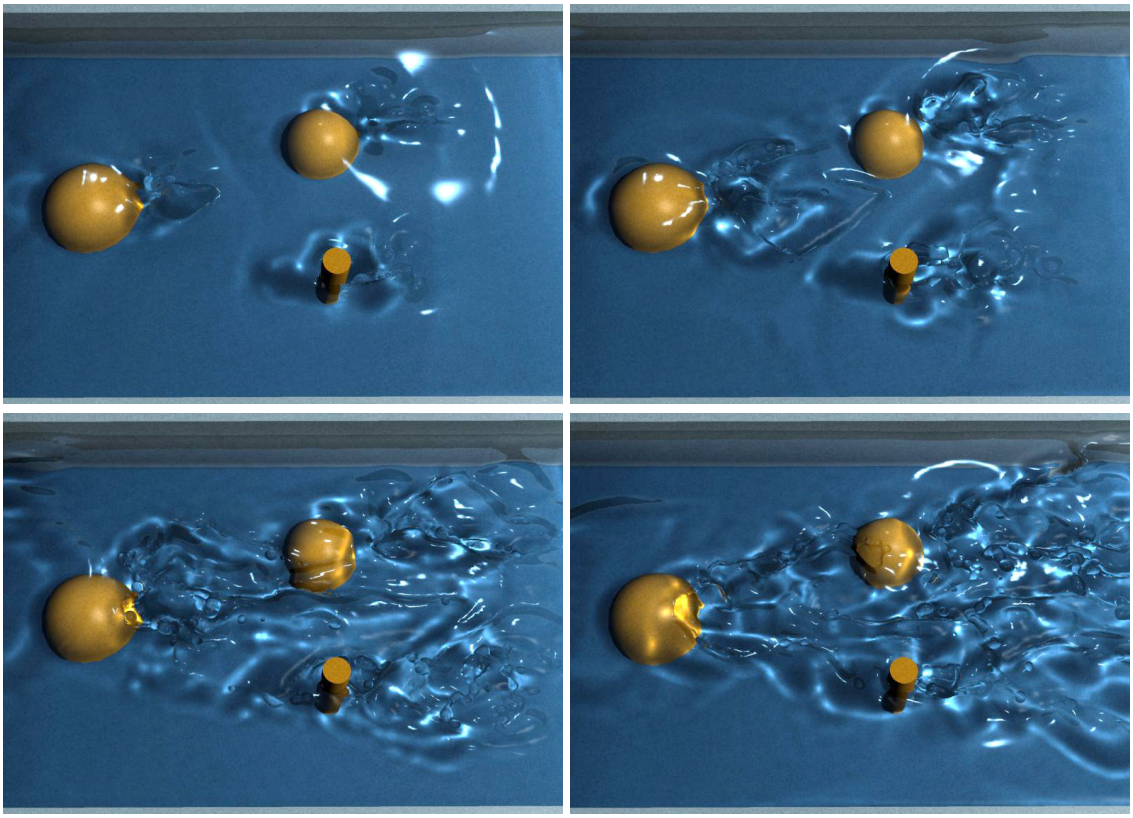


Figure 5.10: *Our algorithm naturally extends to simulations of liquids. Here, we apply our algorithm to a river flow around three obstacles, resulting in turbulent wakes behind them.*

The limitation that motivated vortex particles was that vorticity confinement uniformly amplified vorticity magnitude. Our method, like vortex particles, overcomes this by allowing local modeling of vorticity, including effects like tilting and stretching. However, by modeling the boundary layer and considering the directions of particles vortices with Eq. (5.1), we are able to keep vortex particles from disturbing the bulk flow. This allows us to use vortex particles of larger magnitude than the randomly seeded vortex particles.

Complex Examples Next we consider examples with more complex geometry. Fig. 5.6 shows the simulation of a moving car that is emitting smoke. It can be observed how our model reproduces the dependence of turbulence strength from the cars velocity. As can be seen in the top row of Fig. 5.6, a normal simulation of the same resolution would not resolve any shed vortices at the car's surface. Second, Fig. 5.1 shows a thin whisk geometry stirring

smoke. The boundary layer precomputation was done with a high resolution grid that resolved the whisk's wires, while its subsequent use in a smoke simulation was done on a much coarser grid. The standard grid did not resolve the wires, and only approximate velocity boundary conditions were set, resulting in the fluid slightly following the whisk's motion. Still, our algorithm was able to accurately generate vortices that are produced by its motion.

In Fig. 5.8 we show how our method works in conjunction with static flow fields. In this case we precompute a snapshot image of static flow around the object, and use it to advect the boundary layer, the vortex particles and the smoke densities. This simple form of simulation works without an expensive pressure correction step. Despite the simple underlying setup, we are able to produce complex structures forming in the wake behind the obstacle from the interactions of the vortex particles amongst themselves. Lastly, we demonstrate that our method can be easily extended to free surfaces in Fig. 5.10. Here three obstacles in the liquid produce turbulent wakes behind them. For this simulation, a particle level set [Enright et al., 2002b] was used to represent the liquid's surface. Similar to particles near obstacle walls, we reduce a particle's kernel size once it extends past the liquid phase to avoid non-divergence free velocity fields.

Detailed grid sizes and timings for the examples above can be found in Table 5.1. The performance was measured on an Intel Core i7 CPU with 3.0 GHz. The majority of the time used for our approach (denoted by *ABL time* in Table 5.1) is taken up by the advection of the artificial boundary layer. For the liquid example of Fig. 5.10, the performance is strongly dominated by the particle level set. Overall, we achieve computing times ranging from 10 to 20 seconds per frame on average. An exception is the example with a static flow field, which requires only 1.3 seconds per frame.

5.6 Conclusions

In this chapter, we have presented an algorithm for simulating wall-induced turbulence. By leveraging turbulence modeling and wall flow theory, we are able to precompute turbulence generation based on the obstacle geometry. This precomputed object can then be included in various simulations. During the simulation, we determine transitioning regions and introduce appropriate vortex particles to represent turbulence. The particles are then evolved according to the vortex equations of flow to respect energy conservation and cascading. This yields the ability to efficiently compute physically plausible

simulations of turbulence around rigid objects in a variety of settings. In contrast to other turbulence methods, the model presented can be used for obstacles which are too fine to be resolved on the simulation grid, and it can be applied for free surface flows, a topic that has been barely studied in previous work on turbulence. We do note however, that the model is passive in a sense that turbulence generation from the liquid surface is not handled in the model.

A limitation of our method is that our precomputation currently assumes a rigid object, making it difficult to apply it to deforming objects such as cloth. To resolve this, a RANS solver could be coupled to a normal fluid solver to determine the current shear stresses at the object surface. Alternatively, it may be possible to precompute suitable boundary layer data for deforming objects by making use of data compression schemes. Also, our approach for the precomputation assumes the flow around the object can be described by the translational and rotational velocity components. If the flow around the object varies strongly, e.g., due to strong external forces or due to multiple objects in close vicinity, the resulting confined vorticity can differ from the desired values. Extending our approach to handle this more accurately is interesting future work.

As we modeled the turbulence generation based on wall flow, the method will only generate turbulence from obstacle interaction. Turbulence driven by free-stream effects or buoyancy are not handled yet. In addition, a trade-off of our method is the use of vorticity reconstruction at a higher resolution. While this allows us to go beyond the coarse simulation Nyquist limit and get higher resolution detail (a limitation of the original vortex particle method), it means the domain and object boundaries as well as the free-surface boundary conditions are not as well modeled by the reconstructed high resolution velocity field. The need for a high-resolution field also limits the detail level that can be achieved, as for big scenes with fine detail, memory and computation time for operations on the high-resolution field can easily become the bottleneck.

The method presented in this chapter is specifically designed for the important special case of obstacle-induced turbulence. Many flows, such as plumes or explosions, however involve expansion forces and buoyancy, which are also strong sources of turbulence. In the following chapter, we will introduce a method that is able to represent these free-stream sources of turbulence, but can also handle obstacle source. Using vortex methods on interfaces, we can even avoid the need for a high-resolution grid to represent detail.

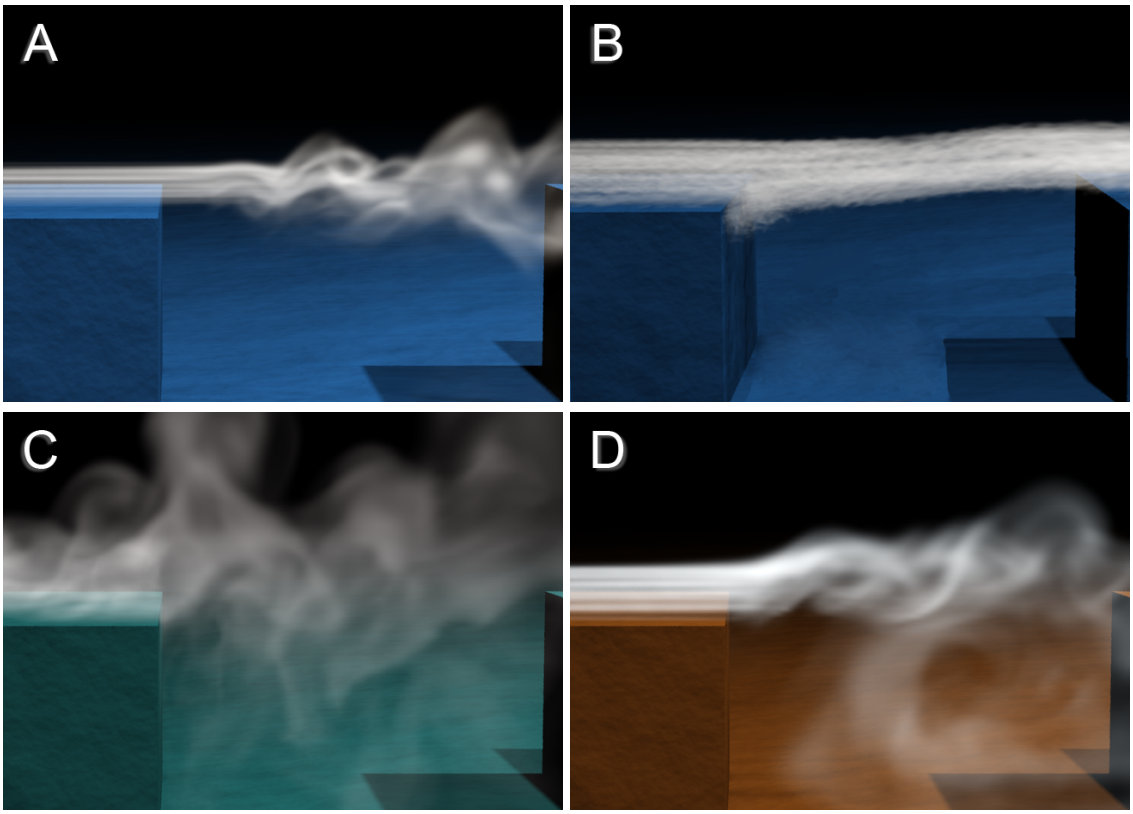


Figure 5.11: *Comparison between a high-resolution reference simulation (A), Wavelet turbulence (B), a simulation with randomly seeded vortex particles along the walls (C), and our method (D). Wavelet turbulence does not predict the turbulence formation, and therefore only amplifies noise. The random vortex particles destroy the overall flow structure, although the number and strength of the vortex particles are similar to those used in our method. Our approach correctly identifies the turbulence being shed off the step, similar to the reference simulation (A), and insert particles into the flow with the correct orientation, strength and seeding position.*

```
1: // Initialize boundary layer
2: for each voxel x on an obstacle boundary do
3:    Find corresponding (x_pre, ω_pre) in precomputed set
4:    // Initialize wall-attached ABL
5:    ω_ABL(x) ← max(ω_ABL(x), ω_pre)
6: end for
7:
8: // Simulate boundary layer separation
9: Advect ω_ABL with the main flow
10:
11: // Seed vortex particles
12: for each voxel x with ω_ABL(x) ≠ 0 do
13:    p_T ← 2 c_P Δt (l_m |ω_ABL(x)| / |U_0|)^2
14:    if random() < p_T then
15:       (Y) ← voxels within particle radius of x
16:       ω_S = Σ_(Y) ω_ABL    // sum within particle radius
17:       ω_ABL(Y) ← 0         // remove vorticity from ABL
18:       Seed particle at x with total vorticity ω_S
19:    end if
20: end for
21:
22: // Vortex particle dynamics
23: Advect vortex particles
24: Merge, split, dissipate vortex particles (§ 5.3.1)
25: Synthesize turbulence (§ 5.3.2)
26:
27: // Standard fluid simulation steps
28: Velocity self-advection, pressure projection etc.
```

Figure 5.12: *Pseudo-code for a main simulation loop including our turbulence model.*

CHAPTER 6

Detail Enhancement on Fluid Interfaces

One of the most visually interesting features of turbulent flows is their complexity. Smoke plumes from volcanoes, explosions or collapsing buildings show detailed motion on scales from several meters down to the millimeter range, and the structure of the developing turbulent eddies is clearly visible at the sharp interface of the thick smoke and the air. At the same time, the thick clouds typically hide everything that is happening further inside the volume. Unfortunately, such scenes are numerically expensive to simulate, and we spend large amounts of computation on detail inside the cloud that will never be visible.

One way of dealing with these complex flow are volumetric turbulence methods, as presented in § 4 and § 5. However, even with a turbulence model the synthesized detail has to be represented in the simulation, and using a volumetric representation resolving the small-scale details requires immense storage capacity.

In this chapter, we therefore chose to explicitly discretize and track only the smoke-air interface. This greatly reduces the amount of information we need to store. In addition, this representation is a very suitable basis for detail synthesis. Instead of unnecessarily calculating detail that is hidden inside the smoke volume, we restrict synthesizing detail purely to the visible smoke interface.

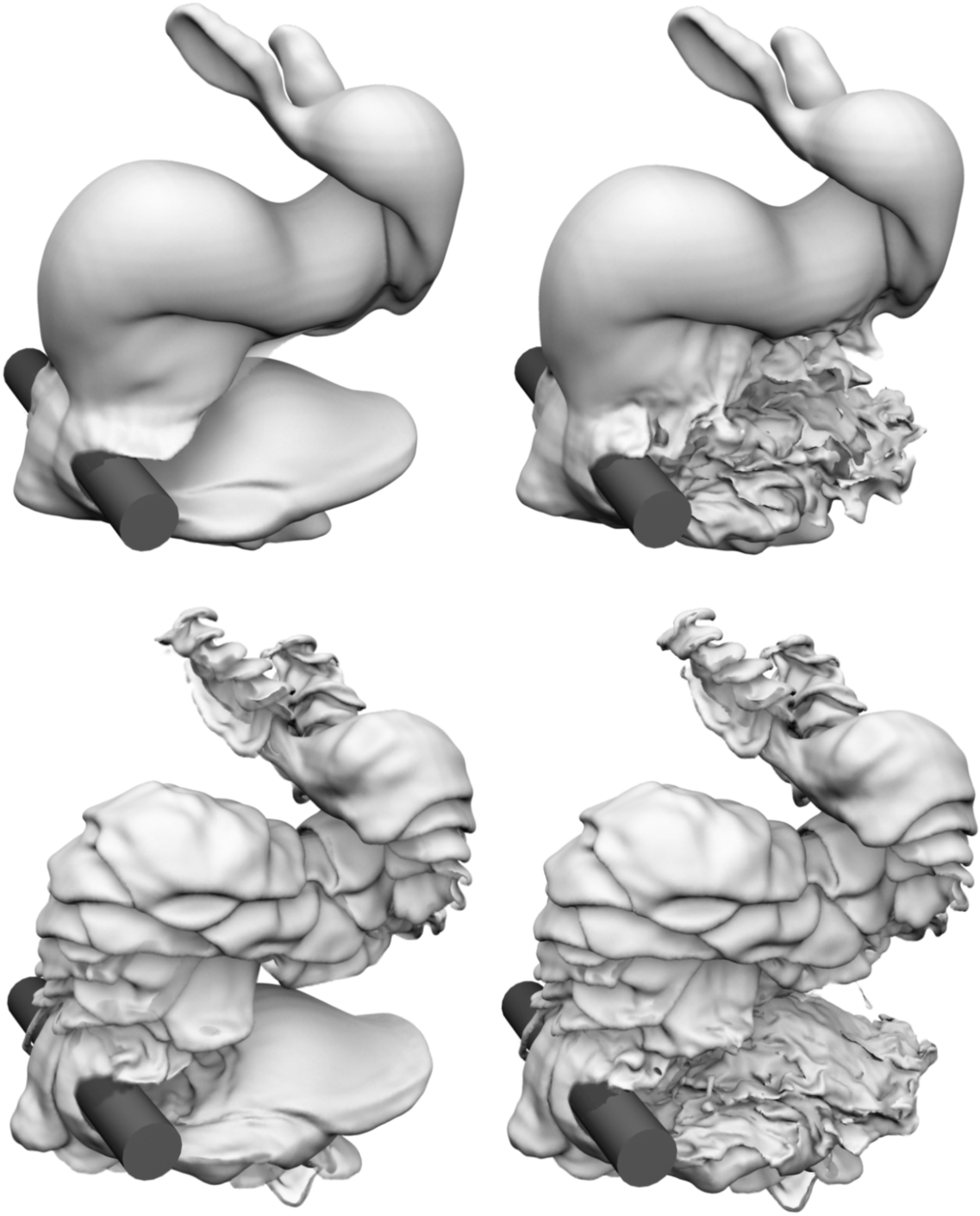


Figure 6.1: *A dense cloud subject to buoyancy forces and interaction with a moving obstacle is simulated. We use a Eulerian solver to compute a base flow, as shown on the left. Small-scale detail is synthesized directly on the interface of the cloud. An adapted turbulence model provides details from obstacle interaction (middle left), while small-scale buoyancy effects are calculated using vortex sheet dynamics (in the middle right). The picture on the right shows the combined model.*

The phenomena mentioned above exhibit another interesting effect: turbulence production in such flows mainly stems from buoyancy, which induces a vortex sheet at the smoke-air interface. This sheet reinforces small-scale surface instabilities, which then develop into turbulence. This means that the transition region where the turbulence is created is clearly visible, and this turbulent onset strongly influences the visible shape of the interface. However, the simulation resolution is typically too limited to directly capture these small-scale buoyancy effects. Furthermore, most turbulence models assume fully-developed homogeneous turbulence, which means they are valid inside the bulk smoke volume, but not at the interface. Here, the turbulence generation process is highly anisotropic and model-dependent in nature. This means it is not well described using the statistical approaches that are the basis for most turbulence methods.

Our method addresses this problem by directly tracking the vortex sheet at the smoke-air interface. This allows us to compute buoyancy effects at scales independent of an underlying grid, and accurately model the turbulence generation process due to buoyancy. While vorticity-based methods are well-suited to describe turbulence formation, correct handling of obstacle boundaries is very difficult. Our model therefore handles basic interaction with static or moving obstacles using a Eulerian solver, and tracks the obstacle-induced turbulence with a model specifically tailored to our needs. We will ensure that the turbulence model for obstacles is orthogonal to our buoyancy approach, which makes it possible to use both in combination or separately as needed.

To summarize, we propose an algorithm with the following contributions:

- A *local evaluation scheme* for vortex sheets which allows us to efficiently capture detailed buoyant and obstacle based turbulence effects.
- A *turbulence model for obstacles* that is able to estimate wall-induced turbulence and is orthogonal to buoyancy based turbulence.
- A *mesh resampling* technique for efficiently pruning invisible detail to reduce mesh complexity.

We use an adaptive triangle mesh to simulate non-diffusive smoke surfaces, and couple it to an Eulerian solver which captures the large-scale motion of the flow. We will demonstrate that this representation is very suitable for vorticity based methods and that it produces highly detailed visuals efficiently.

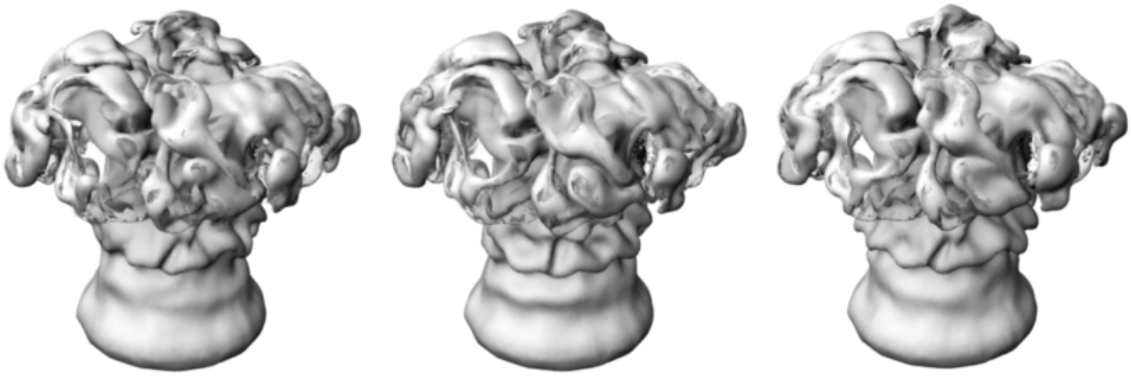


Figure 6.2: *A buoyant plume is simulated without evaluation cutoff (left), with a cutoff of 10 cells (middle) and 5 cells (right, our default setting). While details are different due to accumulation of small differences over time, the visual quality is comparable.*

6.1 Vortex Sheet Methods

Fluid solvers in graphics typically use the velocity formulation of the NS equations to obtain the fluid motion. For dealing with turbulence, however, the vorticity formulation of the NS equations is often advantageous. In general, the evolution of vorticity can be described with the vorticity equation Eq. (3.4). For this method, we will focus on plumes with a sharp density interface, which is a good approximation model for e.g. heavy smoke plumes, or two liquids with different densities. Under the influence of buoyancy or external forces, a thin sheet of vorticity forms at this interface. In our model, we will not track the volumetric velocity or vorticity field, but represent this interface vorticity, or *vortex sheet*, on a surface mesh. The theory of vortex sheets is introduced in § 3.1.2.

Most commonly, the vorticity in vortex sheets is expressed via the vortex sheet strength vector $\boldsymbol{\gamma}$. If we formulate the vorticity equation using $\boldsymbol{\gamma}$, we obtain the evolution Eq. (3.15), with terms for advection, vortex stretching, elongation and baroclinity. By integrating this equation we would be able to calculate the full dynamics of a buoyant plume. As the evolution equation contains operators which are hard to express on a surface representation, this is however not trivial. We therefore also make use of another expression of vorticity, namely the circulation. For vortex sheets, these representations are equivalent and can be converted as explained in § 3.1.2. As some operations are formulated easier in a circulation notation than for vortex strengths, and vice versa, we can simplify the evolution equations by switching between representations. This procedure will be explained in the following paragraph.

Our Model To solve the vorticity dynamics equations, it is necessary to have a discretization of the interface. For this we use a mesh consisting of triangles, where each triangle i has a corresponding vortex strength $\boldsymbol{\gamma}_i$. As we want to make use of the filament representation, too, the three circulation numbers are stored for each triangle in addition to the vortex strength. These circulation numbers $\Gamma_{1\ldots3}$ define a rotation around the triangle's edges $\mathbf{e}_{1\ldots3}$.

As we are interested in buoyant effects, we apply the baroclinic source term in vortex sheet strength notation for each time step.

$$\frac{\partial \boldsymbol{\gamma}}{\partial t} = -2\beta_A \, \hat{\mathbf{n}} \times \mathbf{g} \quad . \tag{6.1}$$

We now recall the fact that the vortex stretching and elongation terms of the evolution equation are implicitly handled in circulation notation, and vanish from the equation. Before evaluating the advection of out surface mesh, we therefore switch to circulation notation

$$\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \end{pmatrix} = A \begin{pmatrix} \boldsymbol{\gamma} \\ 0 \end{pmatrix} \tag{6.2}$$

and return to vortex strength notation afterwards

$$\boldsymbol{\gamma} = \frac{1}{A} \sum_{i=1}^{3} \Gamma_i \, \mathbf{e}_i \quad . \tag{6.3}$$

Using this process, we can avoid the calculation of these operators altogether. We now have taken care of all terms in the vorticity equation. For evaluation of the advection term, we however still need velocity information. This can be integrated from the vortex strength values using the Biot-Savart law, as explained in § 3.1.2. If we look at the integration equation

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int \boldsymbol{\gamma}(\mathbf{x}') \times \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} d\mathbf{x}' \quad . \tag{6.4}$$

we however note that such an evaluation is numerically very complex. For each mesh node, we need to integrate over all triangles, which results in $\mathcal{O}(n^2)$ complexity. As we want to simulate very detailed meshes with millions of triangles, this is prohibitively expensive. Even more importantly, we note that so far, we are only able to obtain the dynamics prescribed by ideal buoyancy in free space. Most practical scenes, however, have a nontrivial underlying flow due to interaction with obstacles and boundary conditions.

In the next section, we therefore introduce a local evaluation scheme, which resolves this issues.

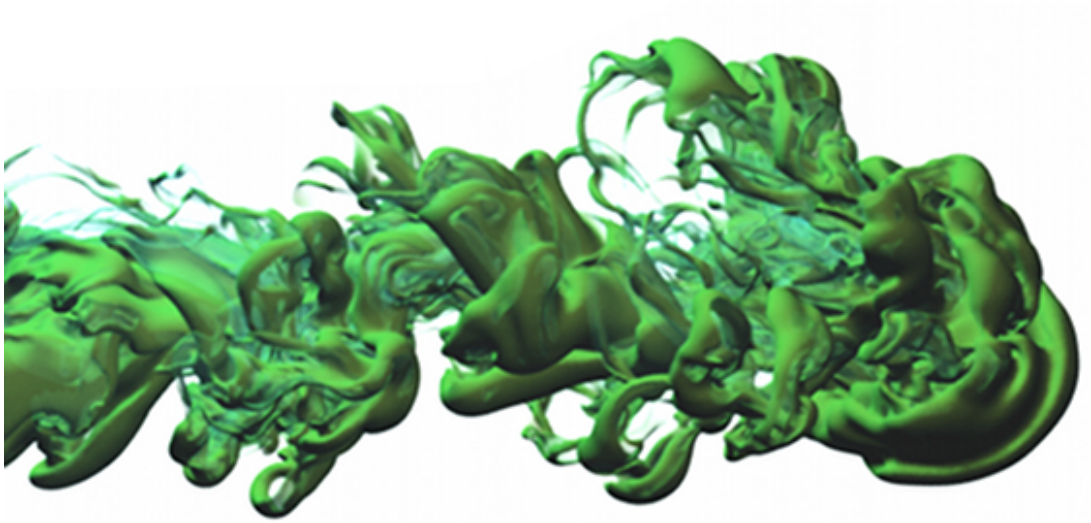


Figure 6.3: *We simulate the dynamics of a dense fluid in water with pulsed inflow conditions. The buoyancy leads to complex surfaces in the downstream region to the right.*

6.1.1 Local evaluation

In our local evaluation model, we split the simulation into two parts: first, a Eulerian solver which computes a consistent flow field from obstacle interaction, inflows, and the large-scale effects of buoyancy. Second, a surface mesh which is used for front tracking of the smoke cloud and the simulation of detail due to small-scale buoyancy effects and obstacle turbulence.

For computation of the large-scale flow, we use a standard grid-based solver [Stam, 1999] with second order semi-Lagrangian advection as described in Selle et al. [2008]. Our vortex sheet approach enables us to use low grid resolutions, as details will be computed directly on the Lagrangian mesh. In the grid-based solver, a density field is tracked which is then used to compute coarse-scale buoyancy forces on the velocity field.

Evaluation of the small-scale buoyancy effects is performed using the vorticity of the mesh. To avoid duplication of buoyancy forces between grid and mesh, we remove the large-scale component of the baroclinic vorticity from the mesh. We first apply a Gaussian smoothing kernel on the vortex sheet strength $\boldsymbol{\gamma}$. The kernel width σ is set to match the grid cell width Δx to obtain the smoothed, grid-scale vortex strength component $\bar{\boldsymbol{\gamma}}$. The difference $\boldsymbol{\gamma}' = \boldsymbol{\gamma} - \bar{\boldsymbol{\gamma}}$ now represents the details below grid scale, which are evaluated on the mesh.

By removing the mean only the high-frequency variations $\boldsymbol{\gamma}'$ remain, whose

1: // *Grid-based Fluid solver*
2: Semi-Lagrangian density and velocity advection
3: Add grid-based buoyancy
4: Pressure projection
5:
6: // *Turbulence model*
7: Compute production: $\mathcal{P}_{wall} = 2\nu_T|\nabla \times \mathbf{U} - \boldsymbol{\omega}_g|^2$
8: Update $\boldsymbol{\omega}_g$ based on Eq. (6.11) and advect
9: Update k, ε based on Eq. (6.8) and advect
10:
11: // *Mesh dynamics*
12: Integrate baroclinity: $\boldsymbol{\gamma}_i \leftarrow \boldsymbol{\gamma}_i - \Delta t\, 2\beta_A\, \hat{\mathbf{n}} \times \mathbf{g}$
13: Compute Gaussian filtered vortex strengths $\bar{\boldsymbol{\gamma}}_i$
14: Small-scale vortex strength: $\boldsymbol{\gamma}'_i \leftarrow \boldsymbol{\gamma}_i - \bar{\boldsymbol{\gamma}}_i$
15:
16: Compute circulations $\Gamma_i \Leftarrow \boldsymbol{\gamma}_i$, Eq. (6.2)
17: **for each** mesh vertex i **do**
18: $\quad \mathbf{u}_i \Leftarrow$ Integrate Eq. (6.7) for sources $\boldsymbol{\gamma}'_i$ within r_C
19: $\quad$ Advect vertex with $\mathbf{u}_i$ and grid velocity field
20: $\quad$ Advect vertex with synthesized curl noise $\mathbf{u}_T = \sqrt{\alpha_S k}\mathbf{y}$
21: **end for**
22: Compute Vortex strengths $\boldsymbol{\gamma}_i \Leftarrow \Gamma_i$, Eq. (6.3)
23:
24: Perform mesh surface smoothing
25: Perform edge collapses and triangle subdivision

Figure 6.4: *Pseudo-code for the simulation loop of our algorithm.*

effect decays very quickly in the far field. This corresponds to the formation of small vortices, which act locally. We are therefore able to introduce a cutoff radius r_C to the evaluation. Only triangles within this radius have to be evaluated in the summation of Eq. (6.7). As we can rely on the grid solver to capture the large scale buoyant motion, the effects of this approximation are negligible. A comparison of a full evaluation versus two different cutoff radii can be seen in Fig. 6.2. As the cutoff approximation introduces small differences which accumulate over time, the resulting surfaces differ. However, the visual quality is comparable for all three simulations, while the processing time is five times faster using $r_C = 5\Delta x$. We use this value for all following simulations with our model. The position update for the mesh nodes is performed based on the Eulerian velocity field, and by applying a per-node velocity update for the small-scale structures, which is described

next. The complete simulation loop for our combined solver is summarized in pseudo code in Fig. 6.4.

6.1.2 Regularization

To obtain the small-scale velocity update for the mesh, Eq. (6.4) is discretized, using the residual vorticity $\boldsymbol{\gamma}'$ as a source. As this equation is singular for points on the interface, we chose to regularize the equation analogous to the vortex blob regularization for vorticity particles [Chorin and Bernard, 1973]

$$\mathbf{u}_{reg}(\mathbf{x}) = \frac{1}{4\pi}\int_S \boldsymbol{\gamma}'(\mathbf{x}') \times f_{reg}(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \tag{6.5}$$

$$f_{reg}(\mathbf{r}) = \frac{\mathbf{r}}{(|\mathbf{r}|^2+\alpha_R^2)^{\frac{3}{2}}} \quad . \tag{6.6}$$

The regularization parameter α_R effectively controls the minimal size of the generated vortices. We therefore set α_R proportional to the mesh resolution, as will be explained in § 6.3.2. To discretize this equation, we use Gaussian quadrature. If $G_j(\mathbf{r})$ is the Gaussian quadrature of f_{reg} for triangle j, Eq. (6.5) becomes

$$\mathbf{u}_i = \frac{1}{4\pi}\sum_{j=1}^{m} A_j \boldsymbol{\gamma}'_j \times G_j(\mathbf{r}_i) \quad , \tag{6.7}$$

for all triangles within the cutoff radius. This means we need to evaluate a sum over all triangles $j = 1 \ldots m$ per mesh node i. In our examples, we use three-point quadrature, and refer the reader to [Cowper, 1973] for details on how to compute the integration weights.

6.2 Wall-based Turbulence Model

The method presented so is able to model buoyancy driven below grid scale, but does not deal with interaction with flow obstacles yet. While the coarse grid solver introduced in the local evaluation scheme provides the large scale interaction of the flow with obstacles, turbulence shed from these interactions is not represented. However, since our mesh representation allows us to evaluate synthesized turbulence directly on the interface, we can employ a turbulence model similar to § 4 for this type of turbulence.

The turbulence model we propose in the following is orthogonal to the buoyancy model of the previous sections, and both models can be used independently or in combination. We first model the spatial and temporal distribution of turbulent kinetic energy k using an *energy transfer model*,

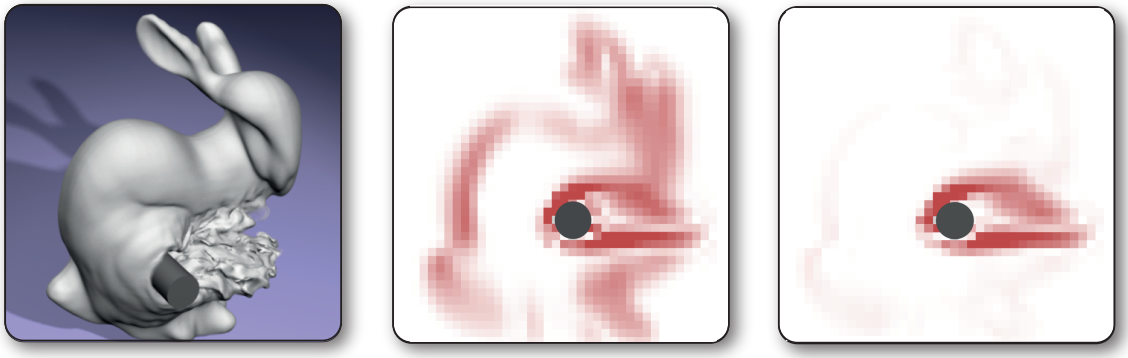


Figure 6.5: *To separate the sources of buoyancy and wall-based turbulence, buoyant vorticity is tracked over time. The total vorticity of a snapshot from Fig. 6.1 is shown in the middle picture, while the difference to the tracked buoyant vorticity is shown to the right. The gray circle marks the position of the cylinder. We observe that despite a small residual halo, our model tracks the area of obstacle influence behind the cylinder very well.*

and then synthesize turbulent detail on the surface using *frequency-matched curl noise*. Below, we will briefly outline the theory used, and explain our modifications. Turbulence modeling is described in more detail in the § 3.3.

6.2.1 Modified Energy Model

We compute the energy dynamics based on the commonly used k–ε model by Launder and Sharma [1974], which models the evolution of the turbulent energy k:

$$\begin{aligned} \frac{\mathrm{D}k}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_k}\nabla k) + \mathcal{P}_{wall} - \varepsilon \\ \frac{\mathrm{D}\varepsilon}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_\varepsilon}\nabla \varepsilon) + \frac{\varepsilon}{k}(C_1\mathcal{P} - C_2\varepsilon) \quad . \end{aligned} \tag{6.8}$$

Details of the model can be found in 3.3.1.

Instead of solving this equation system on the Lagrangian markers as in § 4, we solve it on the coarse grid which is also used for the local evaluation § 6.1.1. The model can also be solved on the high-resolution surface mesh, this did not yield a significant difference in our experiments. The reason for this is that the variables k and ε are averaged properties, and spatially vary smoothly due to turbulent diffusion. The Eulerian approach has the advantage that it is easier to exclude the effects of buoyancy, as discussed below.

The primary interest here is to compute source terms for driving the model. The sources should capture the wall-induced turbulence, but exclude turbulence induced by buoyancy. If we were to directly use k for injecting turbulence we would include the effects of buoyancy twice: once from the k–ε model, and once from the vortex sheet model. In addition, a general turbulence model would not be able to capture the characteristic effects of buoyancy, such as the cloud billowing. We therefore need to guarantee orthogonality of the two methods, by excluding the effects of buoyancy from Eq. (6.8), such that each model can focus on the type of turbulence it is most suitable for. With a strain-based production term that is commonly used for the k–ε model, this would however imply separating the wall induced turbulence from the total one. This is, to the best of our knowledge, not possible for a strain based production. There is, however, an alternative production term $\mathcal{P}_R$ based on rotation. Compared to the strain based measure, it is less accurate for free-stream generation but still captures buoyancy and wall induced turbulence very well. Assuming we have a measure for the current buoyancy-induced turbulence, we can subtract it from $\mathcal{P}_R$ to single out the turbulence induced by obstacles. We have found that using the rotation-based production term from Spalart [1994] and a vorticity based integration of the buoyancy production allows us to do just this.

According to Spalart [1994], the production is given by

$$\mathcal{P}_R = 2\nu_T \sum_{i,j} \Omega_{ij}^2 \tag{6.9}$$

with the rotation tensor Ω_{ij}. We now express its tensor norm in terms of vorticity as $\sum_{i,j} \Omega_{ij}^2 = |\boldsymbol{\omega}_f|^2$. Here $\boldsymbol{\omega}_f$ is simply the vorticity of the grid-based flow field given by $\boldsymbol{\omega}_f = \nabla \times \mathbf{U}$. With $\boldsymbol{\omega}_g$, which denotes the buoyancy induced vorticity strength that we will compute below, we obtain turbulence production for purely wall-generated turbulence using the difference of the two:

$$\mathcal{P}_{wall} = 2\nu_T |\nabla \times \mathbf{U} - \boldsymbol{\omega}_g|^2 \quad . \tag{6.10}$$

For stability, we ensure that $|\nabla \times \mathbf{U}| \geq |\boldsymbol{\omega}_g|$. An example from the simulation of Fig. 6.1 comparing the two vorticity measurements can be found in Fig. 6.5. Finally, we need to compute the accumulated vorticity induced by buoyancy $\boldsymbol{\omega}_g$. Applying the Boussinesq assumption and omitting external forces, we obtain an evolution equation for the buoyant vorticity $\boldsymbol{\omega}_g$ with

$$\frac{\mathrm{D}\boldsymbol{\omega}_g}{\mathrm{d}t} = \boldsymbol{\omega}_g \cdot \nabla \mathbf{u} + \frac{1}{\rho}(\nabla \rho \times \mathbf{g}) \quad . \tag{6.11}$$

We integrate this equation over time on the grid in combination with the k–ε model to obtain the wall based turbulence production $\mathcal{P}_{wall}$ as outlined in

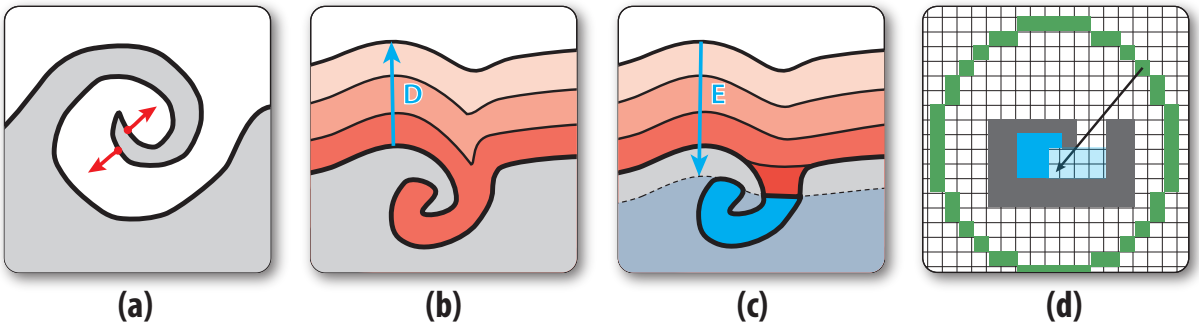


Figure 6.6: *To simplify mesh geometry, we collapse invisible thin sheets. We fist identify candidate nodes in very thin sheets (a). Next, we compute an eroded inside volume on grid in steps (b) and (c). Finally, we check whether these cells are visible with a raycast towards an enclosing sphere (d). All thin sheet nodes in the blue region of (d) are marked for edge collapses.*

Fig. 6.4. Equipped with this production term we compute the spatial distribution of the turbulent kinetic energy k that we use to synthesize turbulent detail on the smoke surface.

6.2.2 Turbulence Synthesis

In contrast to buoyancy induced turbulence, we can synthesize the turbulence triggered by our obstacle-induced turbulence model using K41 theory § 3.4. In this regime energy is mainly scattered from large to small scales, so we can approximate the velocity of the turbulent details using a frequency-matched curl noise texture that is advected through the large-scale velocities, as described in § 3.5.1. Instead of evaluating the turbulence at each cell of a higher resolution grid, we can synthesize it more accurately on the mesh. Each mesh node carries a texture coordinate $\mathbf{q}$ for curl noise texture, and its turbulent kinetic energy k is interpolated from the grid. The additional velocity per node is then given by

$$\mathbf{u}_D(\mathbf{r}) = \nabla \times \sqrt{\alpha_S k}\,\mathbf{N}_\mathbf{f}(\mathbf{r}) \tag{6.12}$$

where $\mathbf{N}_\mathbf{f}$ are the curl noise functions and α_S is a scaling parameter to control turbulence strength. We will demonstrate the interplay of the two turbulence models and their orthogonality in § 6.4.

6.3 Implementation

In this section, details and parameters of our implementation in respect to turbulence estimation, mesh resampling and rendering are specified. To ease

the reproduction of our algorithm, the source code is also publicly available in the MantaFlow project, see § A.3.

6.3.1 Turbulence Model

To solve Eq. (6.8) on the grid, we perform operator splitting as for the Navier-Stokes equations. The advection of k and ε in the PDE system is treated identical to the velocity self-advection using the MacCormack algorithm. The diffusion component $\nabla(\frac{\nu_T}{\sigma_k}\nabla k)$ is expressed using finite differences, with substepping if the CFL condition is violated. To prevent instabilities in the k–ε model for low turbulence intensities we ensure that k and ε are always in a meaningful range where a minimal amount of ambient turbulence is present. Bounds for k are given in terms of turbulence intensity I as $k = \frac{3}{2}{U_0}^2 I^2$, with the characteristic velocity U_0 which is an estimate of the velocity scale in the simulation. We use $I_{min} = 10^{-3}$, $I_{max} = 1$. We found ε is best limited using the equation for the turbulent viscosity ν_T, as this parameter linerly affects production. In our experiments, $\nu_{min} = 10^{-3}$, $\nu_{max} = 5$ are used. As starting parameters for a weakly turbulent initial state we found $\nu_T = 0.1$, $k = 0.1$ to produce stable results.

6.3.2 Mesh Resampling

Due to advection and buoyancy, the mesh will undergo strong deformations. On the other hand, Gaussian smoothing and buoyancy integration rely on a relatively uniform mesh geometry. Therefore, we split and collapse triangle edges to keep all edge lengths l in the range $\Delta l < l < 2\Delta l$, where Δl is the desired minimal edge length. Vortical forces smaller this minimal length would only be visible as a slight noise on the surface. So we use the regularization parameter α_R in Eq. (6.5) to enforce a minimum vortex size larger than Δl. For our example scenes, we chose $\alpha_R = 2\Delta l$. Finally, we apply a small amount of explicit Laplacian smoothing to the mesh [Desbrun et al., 1999], to prevent the accumulation of small-scale noise on the surface.

The vortical motion on the mesh interface creates vortex roll-ups, which lead to the generation of spiral-shaped thin sheets. Since vorticity generation is linked to the surface normal, both sides accumulate almost equal amounts of vorticity, with opposing direction vectors. As the sheets become thinner, the vorticity effect on surrounding nodes therefore becomes smaller and effectively cancels out. Also, many of these thin structures are typically hidden inside the bulk volume of the cloud. Based on these two observations we propose the following algorithm to identify these sheets and remove the

ones that are invisible from the outside. First, we mark nodes on thin sheets, check which of these are far inside volume, and finally perform a visibility test to determine nodes not visible from the outside. The process is visualized in Fig. 6.6.

As a first step, *thin sheet nodes* are identified by checking for a vertex with opposing normal ($\pm$ 20°) within close proximity, i.e. at a distance less than Δl opposing the vertex normal. This can be done efficiently using the grid as acceleration data structure. Next, we identify the volume inside the cloud on the grid. As a coarse representation of the outer hull, we first compute a level set for the mesh. Since triangle size is always well below the size of a grid cell, we can employ a simple and fast method [Kolluri, 2005] to obtain the signed distance function (SDF). We then enlarge and shrink the level set to close small holes and cavities induced by the complex mesh geometry. The level set is enlarged by $D = 4$ cells to compute an outer interface. We rebuild the SDF at a distance $E = -(D + 2)$ from this interface, to obtain a faired volume slightly smaller than the original one. All cells inside this volume are marked as *inside cells*.

As cells in a cavity might still be visible from the outside, we finally compute visibility for the inside cells by performing a raycast towards target points on a sphere enclosing the surface mesh. The cost for these tests is less than 5% for our simulations, as there are typically few cells to be tested. All thin sheet nodes that are located in cells identified as not visible from the outside are marked to be collapsed during the next edge collapse step in line 25 of Fig. 6.4. For the example setup of Fig. 6.8, this method reduces the number of triangles by 32% at the end of the simulation, resulting in an overall speedup of 43%. We note that this reduction based on edge collapses could be improved, e.g., by using methods like [Wojtan et al., 2010], but we have found it to be efficient both in terms of stability as well as performance.

Rendering We use three different methods to render the simulation results.

- For very dense volumes, the mesh could be displayed directly. However, we have found that it is beneficial to add a certain amount of transparency for very thin structures. In the shader, we check the thickness of the volume. If it is above a certain threshold, we render it opaque, otherwise semi-transparently with a transparency proportional to its thickness. We use a threshold of $\Delta x/2$ in our examples.
- To emphasize the detailed structures from the surface vorticity model we can leverage the fact that smoke often concentrates on the vortex sheets [Stock, 2006]. To highlight these surfaces, we modulate the transparency

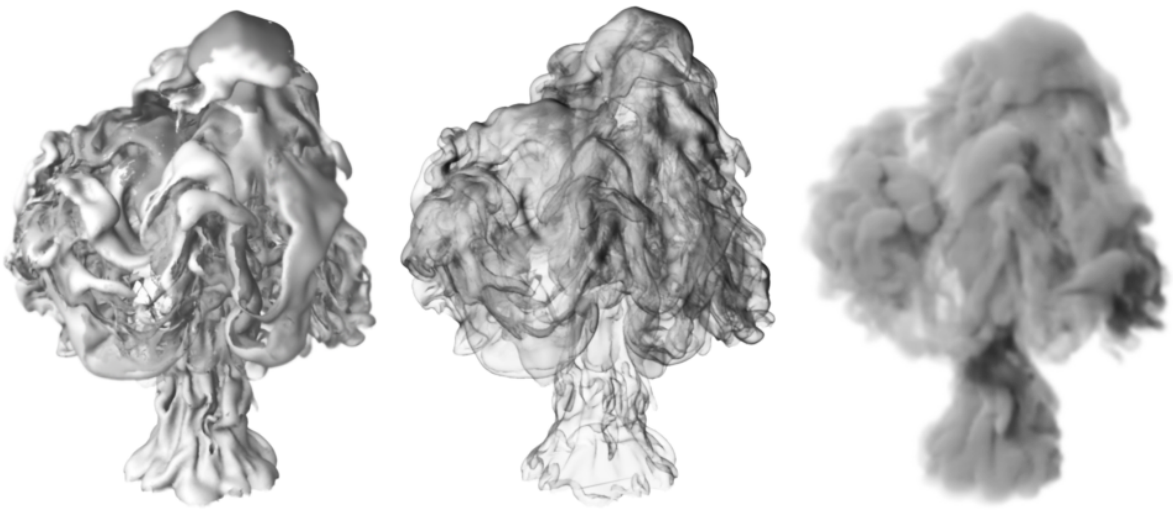


Figure 6.7: *A plume is rendered using semi-transparent rendering (left), wispy smoke rendering (middle) and volume rendering (right). While volume rendering produces the most realistic results for dense plumes, semi-transparent and wispy rendering enhance the visualization of the vortex sheet structure.*

by an approximation term for smoke sheets as given in Funck et al. [von Funck et al., 2008]. To prevent the apparent increase of smoke density by elongation of the mesh, we track the smoke concentration at each triangle during simulation. It is seeded with a constant value at the inflow, and distributed during re-meshing. This per-triangle concentration is multiplied onto the transparency during rendering.

- Lastly, it can be useful to leverage the commonly used volumetric shaders of an existing rendering pipeline. To do this, we project the mesh onto a grid data structure. This density grid might require a high resolution, but is independent of the simulation resolution and only required for rendering.

The effect of these different rendering techniques can be seen, e.g., in Fig. 6.7. For most of the example scenes we have used the semi-transparent shader, the only exception is Fig. 6.9, where we used the volumetric shader.

Performance For high-resolution triangle meshes, the two most costly steps in the simulation loop are applying the Gaussian kernel to the mesh, and integrating Eq. (6.7). However, these operations are simple and do not depend on neighborhood information. Therefore, they are very suitable for parallelization. Using GPU computing with CUDA, we obtained significant speedups of approximately a factor of 10. In the CUDA routine for calculating the velocity update, we use a precomputed hash grid structure to exclude triangles outside the cutoff radius. We note that the complexity can be fur-

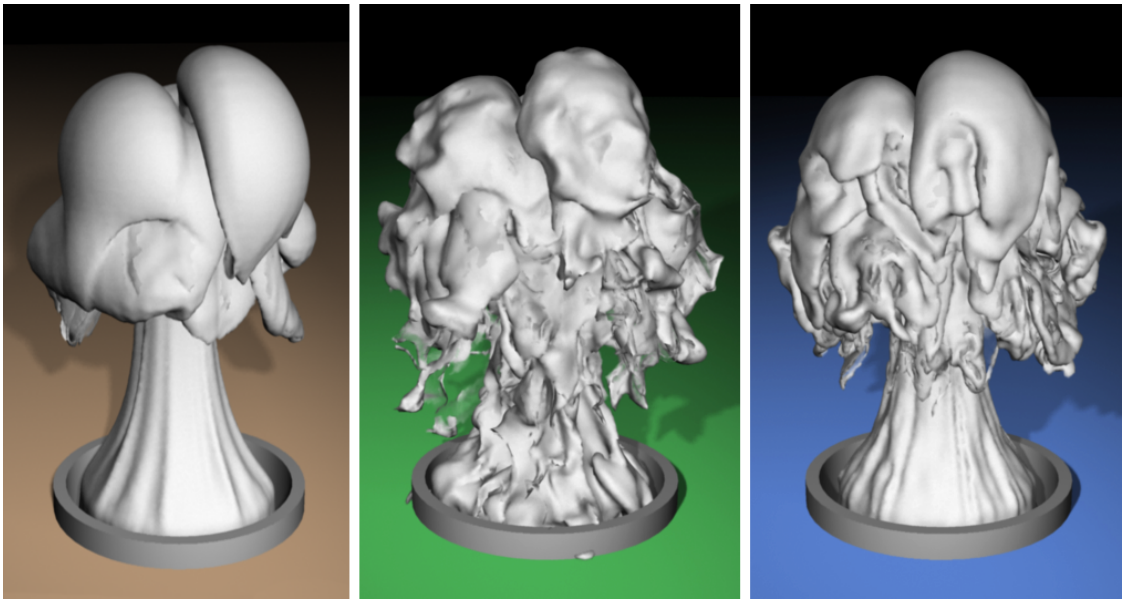


Figure 6.8: *We compare the simulation of a buoyant plume with isotropic turbulence modeling (middle) to our method (right). The base simulation is shown on the left. While isotropic turbulence creates unrealistic surface distortions, the turbulence onset is calculated correctly using our approach.*

ther reduced using Treecodes, e.g. [Qian and Vezza, 2001]. For our example scenes with a few hundred thousand vertices, we however found our simple approach to be sufficient.

6.4 Results

In the following, we demonstrate the properties of our model based on several simulations setups.

Turbulence onset To demonstrate the ability of our vortex sheet dynamics to correctly compute the turbulence onset, we simulated a buoyant smoke plume as shown in Fig. 6.8. The setup uses $64 \times 96 \times 64$ grid cells for the base solver, and a triangle edge length $\Delta l = 0.18\Delta x$. Without artificial disturbing forces, the base flow remains smooth and does not show any turbulent detail. To demonstrate the effect of standard turbulence methods, we synthesize turbulence using vortex particles. The vortex particles are emitted at the inflow and moved along the flow with the smoke plume. For the particles, we use a size and energy distribution based on the Kolmogorov spectrum. This is typically a good assumption for bulk volume flows, as isotropization

Figure 6.9: *In this example scene, an expanding, turbulent smoke front is simulated. The typical cloud billowing is clearly visible in the smoke plume shape. This effect can not be achieved using turbulence synthesis.*

drives the turbulence towards a Kolmogorov spectrum eventually. At the interface, however, the length scales are model-dependent and production is highly anisotropic. This leads to a lack of coherent features using isotropic turbulence methods. Using our method, we observe that the generated detail organically integrates with the large-scale flow.

Eulerian-Lagrangian coupling We demonstrate the generality of our model by simulating two setups with more complex boundary conditions. The first scene, depicted in Fig. 6.9, shows strongly billowing clouds moving through a channel of irregularly shaped obstacles. We simulate an expanding front of smoke with density slightly above air, with a base resolution of $40 \times 40 \times 128$. It can be seen that the flow easily follows the geometry of the scene due to the Eulerian simulation, while our vortex sheet model leads to the development of the typical billowing cloud surfaces. In the second scene, the interaction between water and a heavier liquid is simulated. We use a base solver with

$96 \times 64 \times 64$ grid cells, and pulsed inflow conditions to simulate the injection of multiple drops of fluid. In this case, the temporally changing inflow leads to complex density surfaces developing over time from the buoyant turbulence. Note that the irregular walls of the first, and the pulsed inflow of the second example would be difficult to realize with a simulation based on a pure vorticity formulation.

Wall turbulence In a next example, the interplay between mesh buoyancy and our turbulence model is investigated. To this end, we simulate a plume under the influence of buoyancy and a moving obstacle. Fig. 6.1 shows the orthogonality of the both models: with only the turbulence model activated, we observe detailed structures forming in the wake of the obstacle, while the rest of the flow remains laminar. Once the vortex sheet model is enabled, the mesh shows small-scale deformations with correct orientation due to buoyancy. We show that by combining the two models, we can benefit from both the accurate prediction of source regions by the turbulent energy model, as well as the anisotropic generation of the vortex sheet method. This example exhibits a large number of highly detailed swirls, many of them less than a fifth of a cell in diameter. These surface details are not smeared out despite moving along with the fast and turbulent velocities. Representing this detail during the course of a purely grid-based simulation would require a large amounts of memory, and corresponding amounts of computation for the advection step.

Performance The two most costly steps are applying the Gaussian kernel to the mesh, and integrating Eq. (6.7). Since these operations are simple and do not depend on neighborhood information, we evaluate them on the GPU. This leads to an average time of 10s per frame for the example scenes shown. The majority of this time is spent on the vortex sheet evaluation, i.e. the performance primarily depends on the number of triangles in the mesh. The number of triangles is in turn determined by two factors: the shot length, as triangle numbers typically increase during the course of a simulation, and the re-meshing resolution Δl. The parameter Δl can therefore be used as a means for fine-tuning detail versus performance. The performance numbers and statistics for all scenes can be found in Table 6.1, where *base only* refers to the plume simulation without a turbulence model.

To evaluate the performance of our approach compared to the Vortex-in-Cell (VIC) scheme used, e.g., in Stock et al. [2008], we have simulated the buoyancy only setup shown in Fig. 6.10. We measured computation times up to 19 times faster using our algorithm. We note that our VIC implementation

Setup	Grid res.	#tris mio.	$\Delta l / \Delta x$	Mesh [s]	Grid [s]
Bunny Fig.6.1	$64 \times 64 \times 64$	0.9 / 2.6	0.2	9 / 33	0.6
Water Fig.6.3	$96 \times 64 \times 64$	0.8 / 3.2	0.15	12 / 40	1.3
Plume Fig.6.8	$64 \times 96 \times 64$	0.6 / 2.3	0.18	7 / 22	0.6
- w/o cutoff	$64 \times 96 \times 64$	0.6 / 2.4	0.18	36 / 101	0.5
- base only	$64 \times 96 \times 64$	0.2 / 0.8	0.18	1 / 6	0.5
- vortex part.	$64 \times 96 \times 64$	0.4 / 1.5	0.18	5 / 16	0.6
Street Fig.6.9	$40 \times 40 \times 128$	1.0 / 1.8	0.2	11 / 41	0.9
Duck Fig.6.10	$64 \times 96 \times 64$	0.8 / 3.1	0.2	8 / 30	0.4
- VIC 64	$64 \times 96 \times 64$	0.1 / 0.3	0.2	0.2 / 0.4	6 / 16
- VIC 256	$256 \times 384 \times 256$	0.8 / 3.8	"	4 / 11	156 / 350

Table 6.1: *Performance measurements for our simulation runs. Timings are mean runtime per frame. Two values with a "/" denote the mean and maximum values, respectively.* Grid *refers to all Eulerian operations, while* Mesh *represents vortex sheet dynamics. All simulations were run on a workstation with an Intel Core i7 CPU, a NVidia GTX 580 graphics card and 8GB of RAM.*

uses OpenMP, but no GPU acceleration, as we found that the algorithm is non-trivial to port to the GPU. We still think that this comparison is a good indicator of the complexity of the algorithms, despite the fact that both implementations are not optimized to their full extent.

6.5 Conclusion

In this chapter, a novel algorithm for simulating buoyant, turbulent smoke plumes was presented. A Lagrangian surface mesh is used to track the smoke/air interface. On this mesh, we solve the vortex sheet dynamics, and couple it to a low-resolution Eulerian fluid solver. This allows us to correctly simulate the turbulence generation process on the interface, which is important for visual coherency. On the other hand, the coupling with Eulerian large-scale dynamics allows us to evaluate the update of the velocity in a purely local fashion. This greatly reduces the complexity, and enables the efficient simulation of detailed plumes with non-trivial static boundaries or moving obstacles. In addition, we have proposed an orthogonal turbulence model for capturing turbulence production from obstacles.

A limitation of this approach is that it can lead to meshes with large numbers of triangles. Due to re-meshing, the number of triangles will increase over time in turbulent regions for long simulation times. Although our resampling

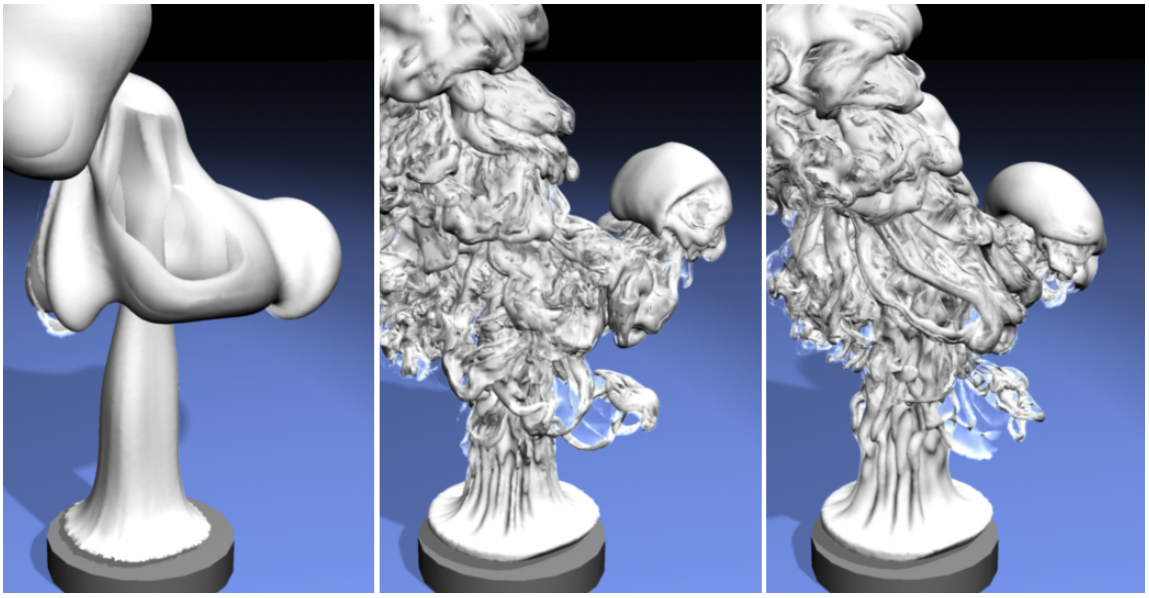


Figure 6.10: *We compare our method to Vortex-in-Cell integration. Our approach (middle) produces similar results as VIC on a 256 grid (right), while being 19 times faster. On the other hand, VIC with a resolution of 64 (left) has a comparable runtime to our method, but exhibits significantly less detail.*

approach reduces the complexity of the meshes, more aggressive approaches are an interesting topic for future work. In addition, accumulated integration errors and re-meshing operations can lead to self-intersecting surfaces. Our method is naturally not well-suited for diffuse, hazy smoke. It would however be very interesting to combine our approach with a lower-resolution volumetric density representation. Sharp, detailed interfaces could then be tracked with our method, while the developing diffuse haze around the dense cloud could be represented on the volumetric grid. It would also be possible to add further detail based on the texture coordinates of the mesh, as we have a temporally coherent discretization of the surface over time.

CHAPTER 7

Conclusion

In this chapter we summarize and discuss the principal contributions of the methods introduced in the previous chapters and suggest directions for future investigations.

7.1 Discussion

In this thesis, we presented three different approaches for enhancing detail of turbulent fluid simulations for Computer Graphics. We leveraged a variety of flow representations and concepts from Computational Fluid Dynamics to analyze flow structure and measure flow properties. This allowed us to obtain deeper insights into turbulence dynamics, with focus on the breakdown of coherent flow features and the formation of turbulence. The understanding of turbulent flow processes is the key element in this thesis, as it enables us to devise methods to accurately model these processes, and use it to augment simulation with generated detail.

In chapter 4, we presented a method to simulate highly detailed turbulent fluids at real time. We used a modified turbulence model based on the k–ε model [Launder and Sharma, 1974] to predict turbulence strength in the flow regime, and extended it to include anisotropic effects. Unlike simpler prediction methods such as [Kim et al., 2008b], this model is general and

complex enough to capture the most important turbulent effects from a low-resolution base simulation. Moreover, the complex prediction scheme enables us to avoid feedback of the generated turbulence into the main solver, which is used in e.g. [Selle et al., 2005] for consistency. Avoiding feedback allows for efficient massive parallelization, which is one key element for making our method real-time capable. The second key element for efficiency is that we directly synthesize the turbulent detail onto the particles used for rendering, thereby creating detail exactly where needed. For this synthesis, an anisotropic version of frequency-matched curl noise synthesis was used. In this manner, we have achieved frame-rates between 5 and 30 fps for our million-particle simulations.

While the predictor of the method in § 4 can be considered a general-purpose turbulence model and is applicable in a variety of scenarios, the synthesis does not represent the important transition from laminar to turbulent flow correctly in this model. The reason for this is that the K41 spectrum used assumes fully-developed turbulence, which is not given in transition. But more fundamentally, all classical turbulence methods based on statistical synthesis cannot represent this process, as it involves the breakdown of coherent structures which is hard to represent in a meaningful way for a statistical model. Also, the method is based on curl-noise texture synthesis, and therefore shares the limitations in dynamics of this approach. This aspect will be explained in more detail below.

In § 5, we therefore took a different approach to synthesis, by employing a Lagrangian vortex representation for turbulence which uses an extended version of vortex particles [Selle et al., 2005]. This allowed us to represent coherent features with full anisotropy information, and a more dynamic turbulent motion compared to the application of detail textures. To model the turbulent transition after flow obstacles, we directly modeled the breakdown process in the vorticity formulation using artificial boundary layers. We precomputed the boundary layer source terms for the flow obstacle geometry, and tracked the boundary layer strength in the simulation. A turbulence predictor was then used to determine regions of flow instability where the boundary layer breaks down into turbulence. This not only allows for a correct seeding of turbulence, but enables us to calculate turbulence seeding even from geometries thinner than the grid resolution. The vorticity formulation makes it also very easy for animators to modify the generated turbulence for a desired look.

The turbulence methods presented in § 4 and § 5 have a better scaling behavior than regular reference simulations, and are therefore very useful for the simulation of very detailed smoke plumes. However, dense, highly tur-

bulent large-scale phenomena such as volcanic plumes are still out of reach for desktop simulations. Due to the dense smoke, the turbulent detail is clearly visible on the interface, and an immense resolution is required to store the simulated and generated detail which is needed for a convincing representation. Also, the shape of such smoke clouds is defined by cloud billowing, which is an effect of slow turbulence transition from buoyancy. As in the case of obstacle-induced turbulence, this process cannot be represented using statistical turbulence synthesis.

On the other hand, the dynamics inside dense plumes are often not visible. In § 6, we therefore developed a model which is not applied to volumetric data, but is able to operate only on the visible interface of smoke plumes. This reduces the dimensionality of the problem and makes the method very efficient for large volumes of smoke. As in § 5, we used a vorticity representation for the synthesis of detail. As we modeled the buoyancy source terms on this vortex sheet representation, we have obtained cloud billowing effects, which is not possible using any of the previous detail enhancement techniques. This approach is orthogonal to bulk turbulence models and can also be combined with a modified version of our model in § 4. In this way we were able to achieve highly detailed simulations of large-scale turbulent plumes efficiently.

Significance While the three methods presented use different mechanics and focus on different processes, their general structure is quite similar. A prediction step models a flow process, such as the formation of a certain type of turbulence, and quantifies flow properties in a statistical sense. This information is used to drive a synthesis step to generate synthetic detail which adheres to the predicted statistics. Finally, the generated detail is used to augment a low-resolution base simulation.

This effectively creates a separation of scales. The larger scales are directly simulated, while the smaller scales are approximated using synthetic detail. By modeling complex turbulence dynamics such as anisotropic generation and vortex sheet rollup, this thesis offloads much of the complexity of a turbulent flow onto detail prediction and synthesis. This shifts the barrier between the simulation and detail synthesis scales. It allows simulations with lower base resolutions and less reliance on the base solver, which results in higher performance. Equally important, it provides us with semantic annotations to the flow. While a detailed flow obtained from a regular high-resolution simulation is hard to manipulate, our methods make it easy for an animator to manipulate turbulence strength, edit the distribution of vortices or add additional sources of turbulence in the flow. We are also able to adjust

the detail level while retaining the overall large-scale flow, which is hard to achieve in a regular simulation due to the chaotic nature of fluids. We see this work as a step forward towards higher-level representation of fluid flows, which will enable new applications in flow analysis and stylistic control.

7.2 Application Guidelines

The new techniques developed in this thesis together with previous methods summarized in § 3 are best seen as a toolbox of methods for simulating complex, turbulent flows. Many of the elements can be interchanged, or combined in different ways depending on the requirements. This section provides guidelines for the practical use of these individual components. Most of them are also available as modules in the open source fluid solver *Mantaflow* described in § A.3, which can serve as a framework for experimentation and research in turbulent fluid dynamics.

Turbulence Prediction The simplest turbulence predictors are vorticity [Fedkiw et al., 2001] and wavelet-decomposition of the velocity field [Kim et al., 2008b]. These prediction however only produces meaningful results for strongly forced turbulence, and will fail in most complex cases, especially when using low resolution base solvers. The arguably most useful representation for non-trivial turbulence prediction is TKE, as a vast set of well-proven tools exists for this representation by the means of classical turbulence models. For many use cases, a turbulence predictor based on a complete two-equation model such as the k–ε model § 4.2.1 provides the best trade-off between complexity and prediction power. On the one hand, the simpler incomplete one-equation models require scene-dependent information such as a mixing length, which are hard to specify in the general case. On the other hand, more complex models such as full Reynolds stress transport rarely pay off for Graphics applications. While they provide more prediction power especially for highly anisotropic and transition flows, it is hard to use the information gained in a meaningful way, as accuracy is limited by the statistical synthesis methods.

Turbulence Synthesis The most popular turbulence synthesis method in Graphics is frequency-matched curl noise texture synthesis as described in § 3.5. This is due to their simplicity, efficiency and the fact that they work well in combination with TKE predictors. Instead of representing and simulating turbulence dynamics, only a texture lookup has to be performed, which

makes it the prime choice in methods geared towards real-time such as § 4. However, this method suffers from a number of severe drawbacks. Firstly, the transition between coherent anisotropic structures and the isotropic textures creates visual artifacts. This can be partly alleviated by 2D anisotropy extensions as described in § 4.2.3, but the method is inherently limited in that detail structures cannot easily be edited or aligned to coherent flow features. Therefore, it will always remain disconnected from the base flow. Also, the modulation of the noise texture with the TKE effectively creates divergences, which may be a problem if strong gradients of turbulence intensity exist in the scene. Even more importantly, the detail dynamics is limited by the static nature of texture. Within an octave, there is no interaction between the generated turbulent eddies, which creates an unrealistic frayed-out look especially if no background flow is present.

For small synthesis scales, i.e. using a high resolution base solver to cover the mid-range turbulence, for flows with mostly homogeneous turbulence intensities or for real-time scenarios, curl noise texture synthesis is therefore a good choice. For all other cases it pays off to directly represent turbulence using a vortex representation, such as vortex particles (§ 5), filaments [Weissmann and Pinkall, 2010] or vortex sheets (§ 6). This most often results in more plausible turbulence dynamics, and allows to model more complex turbulent effects such as transition and breakdown. On the downside, it takes more effort to couple these representations to turbulence predictors, and re-meshing can be an issue. To represent strong turbulence, vortex particles are the prime choice, as these flows tend to be less connected and represented most compactly using particle kernels. Filaments are very efficient to cover mid-level turbulence, and are also useful for the modeling of transition effects. For interface effects, vortex sheets are most efficient. They are also the suited best for baroclinity-driven effects, such as cloud billowing which is hard to model in other representations.

7.3 Future work

The work introduced in this thesis opens up various possibilities for further research. While turbulence prediction has become sufficiently mature for most applications, we believe many interesting research opportunities remain in the area of turbulence representation and synthesis, which are the bottleneck for the methods presented in this thesis.

Noise Texture Synthesis While synthesis using noise textures is very fast, it tends to produce unrealistic dynamics. Since the synthesized velocity does not interact with itself, the generated field remains largely static. Removing this limitation by animating the noise texture would therefore improve the realism of this synthesis approach. However, memory limitations make it hard to store an animated 3D noise texture, and the dynamic animation of the texture at run-time is computationally expensive, which precludes its use for the very scenarios the method is useful for. To improve the dynamics, it should be possible to use a number of narrow-band textures instead of wide-bandwidth noise textures. Each detail velocity lookup would then involve a set of texture coordinates, which causes the high-frequency textures to be advected in the lower-frequency ones. Alternatively, texture animation at run-time could be feasible if the information would not be stored as a Eulerian velocity field, but in a sparser parameterization, e.g. as Lagrangian vector potential elements.

Another field of future research is the behavior of detail texture synthesis over TKE gradients, as the direct modulation of the detail field with the TKE causes divergences in the velocity field. A solution for this problem could be to replace the local lookup with a global synthesis of the detail velocity field incorporating turbulence intensity, using e.g. wavelet functions. The challenge is to perform this global synthesis in an efficient manner.

Lagrangian Representations Lagrangian vorticity primitives are a powerful means to represent and synthesize turbulence. As explained in § 3.1.2, the different primitive types are suitable for representing different types of turbulence. While sheets or filaments are appropriate to simulate turbulence breakdown, fully-developed turbulence is best represented using vortex particles. It would therefore be interesting to combine the various primitives in a simulation, and use them where most appropriate. The main challenges include reconstructing connectivity at the transitions, and ensuring continuity in the velocity field when exchanging primitives, as otherwise coherent flow features will become unstable by the transition.

It is also possible to enhance the capabilities of vorticity primitives by tracking additional parameters. Viscous diffusion could e.g. be introduced by storing a diffuse radius at each primitive, similar to the work of [Leonard, 1980]. This is especially useful for the vortex sheets method § 6, as it allows to model the development of hazy smoke around the cloud interface by diffusive processes.

Adaptivity Finally, we note that the complete separation between detail and low-resolution base solver of methods such as § 3 opens up new possibilities for adaptive methods. While it is hard to change particle numbers in e.g. SPH simulations, since this will create pressure waves, changing the particle resolution in § 3 is unproblematic, as changes are not fed back into the base solver. This allows the development of LOD techniques which adjust particle resolution depending on distance to the camera, or decrease particle density in invisible regions inside e.g. a smoke plume. The challenge is then to recreate shape information when up-sampling a previously down-sampled region.

APPENDIX A

Appendix

A.1 Notation

This section reviews the notation employed throughout the thesis.

Operators and Functions

$\delta(\cdot)$ ..... Dirac's delta function
δ_{ij} ..... Kronecker delta function
$\times$ ..... Vector cross product
$\cdot$ ..... Vector dot product
∇ ..... Gradient / Divergence operator
$\nabla \times \cdot$ ..... Curl operator
∇^2 ..... Laplace operator
$\int_L$ ..... Line integral
$\int_S$ ..... Surface integral
$\langle \cdot \rangle$ ..... Averaging operator
$|| \cdot ||$ ..... Euclidean norm

Scalar parameters

k ..... Turbulent kinetic energy $k = \frac{1}{2}\langle \mathbf{u}' \cdot \mathbf{u}' \rangle$
c_p ..... turbulence transition parameter
l_m ..... Mixing length
p ..... Pressure
p_T ..... Probability density for turbulence transition
r ..... Radius
t ..... Time
s ..... Curve length
Δx ..... Cell width
A ..... Triangle area
C_x ..... Turbulence model constants
E ..... Energy distribution
I ..... Turbulence intensity
L ..... Length scale
N_f ..... Noise field
Re ..... Reynolds number $\mathrm{Re} = \nu L / v$
V ..... Volume
ε ..... Turbulent energy dissipation
$\mathcal{P}$ ..... Turbulent production
α_S ..... Turbulence detail strength coefficient
β_A ..... Atwood ratio $\beta_A = (\rho_1 - \rho_2)/(\rho_1 + \rho_2)$
κ ..... Wavenumber
ρ ..... Fluid density
ν ..... Viscosity
ν_T ..... Turbulent viscosity
σ ..... Kernel width
Γ ..... Circulation number

Vector-valued parameters

$\mathbf{c}$ ..... Curl noise field
$\mathbf{e}$ ..... Edge vector
$\mathbf{g}$ ..... Gravitation
$\mathbf{k}_A$ ..... Anisotropic kinetic energy
$\mathbf{n}$ ..... Surface normal
$\mathbf{q}$ ..... Texture coordinate
$\mathbf{u}$ ..... Velocity
$\mathbf{u}'$ ..... Fluctuating turbulent velocity
$\mathbf{u}_\Psi$ ..... Divergence-free velocity component
$\mathbf{u}_\Phi$ ..... Curl-free velocity component
$\mathbf{u}_D$ ..... Velocity of the synthesized detail
$\mathbf{t}$ ..... Tangent vector
$\mathbf{x}$ ..... Position in space
$\mathbf{F}$ ..... External force
$\mathbf{N_f}$ ..... Vector noise field
$\mathbf{P}_A$ ..... Anisotropic production
$\bar{\mathbf{U}}$ ..... Averaged velocity field
$\mathbf{U}_0$ ..... Reference velocity, e.g. inflow speed
$\boldsymbol{\omega}$ ..... Vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$
$\boldsymbol{\gamma}$ ..... Vortex sheet strength
$\bar{\boldsymbol{\gamma}}$ ..... Filtered vortex sheet strength
$\boldsymbol{\gamma}'$ ..... Detail vortex sheet strength
$\boldsymbol{\Psi}$ ..... Vector potential

Tensors

a ..... Anisotropic Reynolds stress tensor $a_{ij} = \tau_{ij} - \frac{2}{3}k\delta_{ij}$
I ..... Identity operator
P ..... Projection operator $\mathbf{P} = \mathbf{I} - \mathbf{n}\,\mathbf{n}$
S ..... Strain tensor $S_{ij} = \frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})$
T ..... Turbulence transport tensor (third-rank)
$\mathcal{P}$ ..... Turbulence production tensor
$\mathcal{R}$ ..... Turbulence redistribution tensor
ε ..... Turbulence dissipation tensor
τ ..... Reynolds stress tensor $\tau_{ij} = \langle u_i' u_j' \rangle$
Ω ..... Rotation tensor $\Omega_{ij} = \frac{1}{2}(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i})$

A.2 Glossary

ABL ..... Artificial boundary layer
CFD ..... Computational fluid dynamics
CFL ..... Courant-Friedrichs-Lewy condition
FDM ..... Finite difference method
FLIP ..... Fluid implicit particle
FEM ..... Finite element method
FFM ..... Fast multipole method
FVM ..... Finite volume method
GPGPU ..... General purpose graphics processing unit
GPU ..... Graphics processing unit
K41 ..... [Kolmogorov, 1941]
LES ..... Large eddie simulation
LOD ..... Level of detail
MAC ..... Marker and cell
NS ..... Navier-Stokes
PDE ..... Partial differential equation
PIC ..... Particle in cell
RANS ..... Reynolds-averaged Navier Stokes
SPH ..... Smoothed particle hydrodynamics
TKE ..... Turbulent kinetic energy
VIC ..... Vortex in cell

A.3 Software

Most of the methods described in the previous chapters have been implemented in *Mantaflow*. *Mantaflow* is an open-source fluid solver framework, which was developed as part of this thesis. It aims at making it easy for researchers to implement new algorithms and experiment with new concepts in the context of fluid simulation. The output can be visualized during runtime using an integrated GUI or rendered in Maya using a plugin.

A simulation in *Mantaflow* is set up using a scene-definition file written in Python. This is a high-level description of both scene geometry, data flow and the simulation loop, similar to pseudo code given in many fluid research papers. Fig. A.1 shows an example scene definition file for the simulation of a rising plume. *Mantaflow* comes with many useful classes and plugins, so standard simulation tasks can be performed simply by editing scene definition files.

As the framework is geared towards research, a design goal was to make it easy to extend for new algorithms and data types. The functions and statements called from in the scene definition file are implemented in C++ for performance reasons. Classes such as `RealGrid` are implemented as (templated) C++ classes, while plugins such as `solvePressure` are free C++ functions. A custom preprocessor automatically generates the glue code to keep the definition of new plugins and classes simple. A very simple extension plugin function is shown in Fig. A.2, which multiplies a real grid (e.g. density) with a factor. The `PLUGIN` keyword tells the pre-processor to generate glue code to expose the function to Python. `KERNEL` defines a kernel which is evaluated for each cell of a grid, or each particle in a particle system. Kernels can be reused between plugins and are automatically parallelized using Thread Building Blocks.

This section aimed at providing a short glimpse at how to use this framwork. A short tutorial, documentation and the full sources are available at `http://mantaflow.ethz.ch`.

```
from manta import *

# solver params
res = 64
gs = vec3(res,res,res)
s = Solver(name='main', gridSize = gs)
s.timestep = 1.0

# prepare grids
flags = s.create(FlagGrid)
vel = s.create(MACGrid)
density = s.create(RealGrid)
pressure = s.create(RealGrid)

flags.initDomain()
flags.fillGrid()

# define shape for smoke inflow
smokeSource = s.create(Cylinder, center=gs*vec3(0.5,0.13,0.5),
                                 radius=res*0.14, z=gs*vec3(0, 0.03, 0))

# main loop
for t in range(250):
    smokeSource.applyToGrid(grid=vel, value=velInflow)

    # MacCormack advection
    advectSemiLagrange(flags=flags, vel=vel, grid=density, order=2)
    advectSemiLagrange(flags=flags, vel=vel, grid=vel, order=2)

    setWallBcs(flags=flags, vel=vel)
    addBuoyancy(density=density, vel=vel, gravity=vec3(0,-6e-4,0), flags=flags)

    solvePressure(flags=flags, vel=vel, pressure=pressure)
    setWallBcs(flags=flags, vel=vel)

    s.step()
```

Figure A.1: *A scene definition file for a simple rising smoke plume. First, the Solver and Grid objects are set up. Then, geometry is initialized. The main loop consists of the advection of smoke density and velocity fields, integration of buoyancy forces, and the pressure projection operation.*

```
KERNEL(ijk) KnScaleField(FlagGrid& flags, Grid<Real>& density, Real factor)
{
  if (flags.isFluid(i,j,k))
    density(i,j,k) *= factor;
}

PLUGIN void scaleField(FlagGrid& flags, Grid<Real>& density, Real factor)
{
    KnScaleField(flags, density, factor);
}
```

Figure A.2: *This C++ code snippet defines and registers the simple plugin function scaleField. The plugin function applies a kernel over the density grid, which multiplies the density field with a given Scalar for all fluid cells.*

Bibliography

[Agishtein and Migdal, 1989] M. E. Agishtein and A. A. Migdal. Dynamics of vortex surfaces in three dimensions: Theory and simulation. *Physica D*, 40:91–118, 1989.

[Angelidis and Neyret, 2005] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *ACM SIGGRAPH / EG Symposium on Computer Animation*, 2005.

[Angelidis et al., 2006] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *ACM SIGGRAPH / EG Symposium on Computer Animation*, 2006.

[Aupoix, 2004] B. Aupoix. Modeling of compressibiliy effects in mixing layers. *Journal of Turbulence*, 5, 2004.

[Baldwin and Lomax, 1978] B. S. Baldwin and H. Lomax. Thin Layer Approximation and Algebraic Model for Seperated Turbulent Flows. *American Institute of Aeronautics and Astronautics Journal*, 1978.

[Bargteil et al., 2006] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, and John A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1), 2006.

[Barnat and Pollard, 2012] Alfred Barnat and Nancy S. Pollard. Smoke sheets for

graph-structured vortex filaments. In *Proceedings of the 2012 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12. ACM, 2012.

[Batty et al., 2007] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics*, 26(3):Article 100, 2007.

[Beale and Majda, 1982] J. T. Beale and A. Majda. Vortex methods i: convergence in three dimensions. *Math. Comput.*, 159:1–27, 1982.

[Brady et al., 1998] M. Brady, A. Leonard, and D. I. Pullin. Regularized vortex sheet evolution in three dimensions. *J. Comput. Phys.*, 146:520–545, 1998.

[Bridson et al., 2007] Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM SIGGRAPH papers*, 26(3):Article 46, 2007.

[Bridson, 2008] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[Brochu and Bridson, 2009] T. Brochu and R. Bridson. Animating smoke as a surface. *SCA posters*, 2009.

[Brochu et al., 2010] Tyson Brochu, Christopher Batty, and Robert Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4):47:1–47:9, July 2010.

[Brochu et al., 2012] Tyson Brochu, Todd Keeler, and Robert Bridson. Linear-time smoke animation with vortex sheet meshes. In *ACM SIGGRAPH / EG Symposium on Computer Animation*, 2012.

[Carlson et al., 2004] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:377–384, 2004.

[Chaoat and Schiestel, 2007] B. Chaoat and R. Schiestel. From single-scale turbulence models to multiple-scale and subgrid-scale models by fourier transform. *Theor. and Comp. Fluid Dyn.*, 21(3):201–229, 2007.

[Chen et al., 2011] Fan Chen, Ye Zhao, and Zhi Yuan. Langevin particle: A self-adaptive lagrangian primitive for flow simulation enhancement. *Computer Graphics Forum*, 30(2):435–444, 2011.

[Chentanez and Mueller, 2011] Nuttapong Chentanez and Matthias Mueller. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph.*, 30:82:1–82:10, 2011.

[Chentanez and Müller, 2010] Nuttapong Chentanez and Matthias Müller. Real-time simulation of large bodies of water with small scale details. *Proceedings of the 2010 ACM SIGGRAPH Symposium on Computer Animation*, pages 197–206, 2010.

[Chorin and Bernard, 1973] A. J. Chorin and P. S. Bernard. Discretization of a vortex sheet, with an example of roll-up. *J. Comp. Phys.*, 13:423–429, 1973.

[Chorin, 1981] A. J. Chorin. Estimates of intermittency, spectra and blow-up in developed turbulence. *Comm. on Pure and Applied Math.*, 34:853–866, 1981.

[Chorin, 1996] A. J. Chorin. Microstructure, renormalization and more efficient vortex methods. *ESAIM Proc.*, 1:1–14, 1996.

[Chung and Kim, 1995] M. K. Chung and S. K. Kim. A nonlinear return-to-isotropy model with turbulent fluctuations. *Phys. Fluids*, 7:1425–1436, 1995.

[Cohen et al., 2010] Jonathan Cohen, Sarah Tariq, and Simon Green. Interactive fluid-particle simulation using translating eulerian grids. In *Proceedings of the 2010 SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2010.

[Cook and DeRose, 2005] Robert Cook and Tony DeRose. Wavelet noise. In *Proceedings of ACM SIGGRAPH 2005*, volume 25, 2005.

[Cottet and Koumoutsakos, 1999] Georges-Henri Cottet and Petros Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge Univ. Press, 1999.

[Cottet and Poncet, 2003] G. H. Cottet and P. Poncet. Advances in direct numerical simulations of 3d wall-bounded flows by vortex-in-cell methods. *J. Comput. Phys.*, 193:136–158, 2003.

[Cowper, 1973] G.R. Cowper. Gaussian quadrature formulas for triangles. *Int. J. Num. Methods*, 7(3):405–408, 1973.

[Crane et al., 2007] Keenan Crane, Ignacio Llamas, and Sarah Tariq. *Real Time Simulation and Rendering of 3D Fluids*, chapter 30. Addison-Wesley, 2007.

[Dandois et al., 2007] J. Dandois, E. Garnier, and P. Sagaut. Numerical simulation of active separation control by a synthetic jet. *J. Fluid Mech.*, 574:25–58, 2007.

[de Frutus and Novo, 2001] Javier de Frutus and Julia Novo. A spectral element method for the navier-stokes equations with improved accuracy. *SIAM journal on Numerical Analysis*, 38(3):799–819, 2001.

[Degond and Mas-Gallic, 1989] P. Degond and S. Mas-Gallic. The weighted particle method for convetion-diffusion equations. *Part I: The case of an isotropic viscosity*, 53:485–507, 1989.

[Dehnen, 2002] Walter Dehnen. A hierarchial o(n) force calculation algorithm. *J. Comput. Phys.*, 179:27–42, 2002.

[Desbrun et al., 1999] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proc. SIGGRAPH*, pages 317–324, 1999.

[Driest, 1956] E. R. Van Driest. On turbulent flow near a wall. *J. Aeronaut. Sci.*, 23(11):1007–1011, 1956.

[Durbin, 1993] P. A. Durbin. A reynolds stress model for near-wall turbulence. *J. Fluid Mech.*, 249:465–498, 1993.

[Enright et al., 2002a] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.*, 183:83–116, 2002.

[Enright et al., 2002b] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH*, pages pp. 736–744, 2002.

[Fedkiw et al., 2001] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH*, pages 15–22, 2001.

[Feldman et al., 2005] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. In *Proceedings of ACM SIGGRAPH*, 2005.

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

[Galperin and Orszag, 1993] B. Galperin and S. A. Orszag. *Large Eddy Simulations of Complex Engineering and Geophysical Flows*. Cambridge University Press, 1993.

[Gharakhani, 2003] A. Gharakhani. Application of vrm to les of incompressible flow. *J. Turb.*, 4:4, 2003.

[Gingold and Monaghan, 1977] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. Astron. Soc.*, pages 375–389, 1977.

[Goktekin et al., 2004] 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.

[Guendelman et al., 2003] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 22(3):871–878, 2003.

[Hald, 1979] O. H. Hald. The convergence of vortex methods. *SIAM J. Numer. Anal.*, 32:791–809, 1979.

[Hardy et al., 1976] J. Hardy, O. De Pazzis, and J. Pomeau. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. *Physical Review A*, 13:1949–1960, 1976.

[Harlow and Welch, 1966] F. Harlow and E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluids with free surface. *Physics of Fluids*, 8, 1966.

[Haworth and Jansen, 2000] D. C. Haworth and K. Jansen. Large-eddy simulation on unstructured deforming meshes: towards reciprocating ic engines. *Computers and Fluids*, 29:493–524, 2000.

[Hirt and Nichols, 1981] C. W. Hirt and B. D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comp. Phys*, 39:201–225, 1981.

[Hong and Kim, 2003] J. Hong and C. Kim. Animation of bubbles in liquid. *Proceedings of Eurographics 2003*, 22(3), 2003.

[Horvath and Geiger, 2009] C. Horvath and W. Geiger. Directable, high-resolution simulation of fire on the gpu. *ACM SIGGRAPH papers*, 2009.

[Hsu, 1981] C. Hsu. *A curviliniear-coordinate method for momentum, heat and mass transfer in domains of irregular geometry*. PhD thesis, University of Minnesota, 1981.

[Ikits et al., 2004] M. Ikits, J. Kniss, A. Lefohn, and C. Hanson. *GPU Gems: Programming techniques for real-time Graphics*. Addison Wesley, 2004.

[Irving et al., 2006] G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics*, 25(3):805–811, 2006.

[Jang et al., 2010] Taekwon Jang, Heeyoung Kim, Jinhyuk Bae, Jaewoo Seo, and Junyong Noh. Multilevel vorticity confinement for water turbulence simulation. *Vis. Comput.*, 26(6-8):873–881, June 2010.

[Jiménez and Orland, 1993] Javier Jiménez and Paolo Orland. The rollup of a vortex layer near a wall. *Journal of Fluid Mechanics*, 1993.

[John, 2006] Voker John. On large eddy simulation and variational multiscale methods in the numerical simulation of turbulent incompressible flows. *Applications of Mathematics*, 51:321–353, 2006.

[Jones and Chen, 1994] M. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):pp. 75–84, 1994.

[Kim et al., 2005] ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. Flowfixer: Using BFECC for fluid simulation. In *Proceedings of Eurographics Workshop on Natural Phenomena*, 2005.

[Kim et al., 2008a] Doyub Kim, Oh young Song, and Hyeong-Seok Ko. A semi-lagrangian cip fluid solver without dimensional splitting. *Comput. Graph. Forum (Proc. Eurographics)*, 27(2):467–475, 2008.

[Kim et al., 2008b] Theodore Kim, Nils Thuerey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. *ACM SIGGRAPH Papers*, 27(3):Article 6, Aug 2008.

[Kim et al., 2009] Doyub Kim, Oh-Young Song, and Hyeong-Seok Ko. Stretching and wiggling liquids. *ACM Transactions on Graphics*, 28(5):120, 2009.

[Kim et al., 2012] Doyub Kim, Seung Woo Lee, Oh young Song, and Hyeong-Seok Ko. Baroclinic turbulence with varying density and temperature. *IEEE Transactions on Visualization and Computer Graphics*, 18:1488–1495, 2012.

[Klingner et al., 2006] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH*, 2006.

[Kolluri, 2005] Ravikrishna Kolluri. Provably good moving least squares. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 1008–1018, August 2005.

[Kolmogorov, 1941] A.N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds number. *Dokl. Akad. Nauk SSSR*, 30, 1941.

[Lamorlette and Foster, 2002] Arnauld Lamorlette and Nick Foster. Structural modeling of flames for a production environment. In *Proceedings of ACM SIGGRAPH*, 2002.

[Langtry et al., 2006] R. B. Langtry, F. R. Menter, S. R. Likki, and Y. B. Suzen. A correlation-based transition model using local variables. *J. Turbomach.*, 128:123–143, 2006.

[Launder and Sharma, 1974] B. E. Launder and D. B. Sharma. Applications of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Lett. Heat Mass Transf.*, 1:1031–138, 1974.

[Launder et al., 1975] B. E. Launder, G. J. Reece, and W. Rodi. Progress in the development of a reynolds-stress turbulence closure. *J. Fluid Mech.*, 68:537–566, 1975.

[Le et al., 1997] Hung Le, Parviz Moin, and John Kim. Direct numerical simulation of turbulent flow over a backward-facing step. *J. Fluid Mech.*, 330(01):349–374, 1997.

[Leonard, 1975] A. Leonard. Numerical simulation of interacting, three-dimensional vortex filaments. In *Proceedings of the IV Intl. Conf. on Numerical Meth.*, 1975.

[Leonard, 1980] A. Leonard. Vortex methods for flow simulation. *J. Comput. Phys.*, 37:289–335, 1980.

[Lindsay and Krasny, 2001] K. Lindsay and R. Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *J. Comput. Phys.*, 172:879–907, 2001.

[Losasso et al., 2004] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *Proceedings of ACM SIGGRAPH*, pages 457–462, 2004.

[Lozano et al., 1998] A. Lozano, A. Garc?a-Olivares, and C. Dopazo. The instability growth leading to a liquid sheet breakup. *Phys. Fluids*, 10(9):2188–2197, 1998.

[Marshall and Grant, 1996] J. S. Marshall and J. R. Grant. Penetration of a blade into a vortex core: vorticity response and unsteady blade forces. *J. Fluid Mech.*, 306:83–109, 1996.

[Meng, 1978] J. C. S. Meng. The physics of vortex-ring evolution in a stratified and shearing environment. *J. Fluid Mech.*, 84(3):455–469, 1978.

[Menter and Kuntz, 2005] F. Menter and M. Kuntz. A scale-adaptive simulation model using two-equation models. *AIAA paper 05-1095*, 2005.

[Molemaker et al., 2008] Jeroen Molemaker, Jonathan M. Cohen, Sanjit Patel, and Jonyong Noh. Low viscosity flow simulations for animation. In *ACM SIGGRAPH / EG Symposium on Computer Animation*, pages 9–18, July 2008.

[Monaghan, 2005] J. J. Monaghan. Smoothed particle hydrodynamics. *Rep. Prog. Phys.*, pages 1703–1759, 2005.

[Mullen et al., 2009] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-Preserving Integrators for Fluid Animation. *ACM SIGGRAPH Papers*, 28(3):Article 38, Aug 2009.

[Müller et al., 2005] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. *ACM SIGGRAPH / EG Symposium on Computer Animation*, 2005.

[Narain et al., 2008] Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM SIGGRAPH Asia papers*, page Article 166, 2008.

[Obukhov, 1941] A.M. Obukhov. The spectral energy distribution in a turbulent flow. *Dokl. Akad. Nauk*, 32:22–24, 1941.

[Oden and Wellford, 1972] J. T. Oden and L. C. Wellford. Analysis of viscous flow by the finite element method. *AIAA J.*, 10:1590, 1972.

[Panchev, 1971] S. Panchev. *Random Functions and Turbulence*. Oxford: Pergamon Press, 1971.

[Pao, 1965] Y. H. Pao. Structure of turbulent velocity and scalar fields at large wavenumbers. *Phys. Fluids*, 8:1063–1075, 1965.

[Ploumhans et al., 2002] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows. *J. Comput. Phys.*, 178:427–463, 2002.

[Pope, 1983] S. B. Pope. A lagrangian two-time probability density function equation for inhomogeneous turbulent flows. *Phys. Fluids*, 26:3448–3450, 1983.

[Pope, 2000] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[Prandtl, 1945] L. Prandtl. über ein neues formelsystem für die ausgebildete turbulenz. *Nachr. Akad. Wiss. Göttingen K1*, pages 6–10, 1945.

[Qian and Vezza, 2001] L. Qian and M. Vezza. A vorticity-based method for incompressible unsteady viscous flows. *J. Comput. Phys.*, pages 172:515–542, 2001.

[Rasmussen et al., 2003] Nick Rasmussen, Duc Quang Nguyen, Willi Geiger, and Ronald Fedkiw. Smoke simulation for large scale phenomena. In *Proceedings of ACM SIGGRAPH*, 2003.

[Robinson-Mosher et al., 2008] Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ron Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM SIGGRAPH papers*, 27(3):Article 46, August 2008.

[Rosenhead, 1931] L. Rosenhead. The formation of vorticies from a surface of discontinuity. *Proc. Roy. Soc. London*, 134:170–192, 1931.

[Schechter and Bridson, 2008] Hagit Schechter and Robert Bridson. Evolving subgrid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*, 2008.

[Selle et al., 2005] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *Proceedings of ACM SIGGRAPH*, 24(3):910–914, 2005.

[Selle et al., 2008] Andrew Selle, Ronald Fedkiw, ByungMoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable MacCormack method. *Journal of Scientific Computing*, 2008.

[Shankar and van Dommelen, 1996] S. Shankar and L. van Dommelen. A new diffusion procedure for vortex methods. *J. Comput. Phys.*, 127:88–109, 1996.

[Smagorinsky, 1963] J. Smagorinsky. General circulation experiments with the primitive equations. i. the basic experiment. *Monthly Weather Review*, 1963.

[Smith, 1961] Oliver K. Smith. Eigenvalues of a symmetric 3×3 matrix. *Comm. of the ACM*, 4, 1961.

[Spalart and Allmaras, 1994] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA Paper*, 92:0439, 1994.

[Spalart and Rumsey, 2007] Philippe R. Spalart and Christopher L. Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA Journal*, 45(10), 2007.

[Spalart, 2009] P. R. Spalart. Detached eddy simulation. *Annual Review of Fluid Mechanics*, 41:181–202, 2009.

[Stam and Fiume, 1993] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Proceedings of ACM SIGGRAPH*, 1993.

[Stam, 1999] Jos Stam. Stable fluids. In *Proceedings of ACM SIGGRAPH*, 1999.

[Stock et al., 2008] M. Stock, W.J.A. Dahm, and G. Tryggvason. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. *J. Comp. Phys.*, 227:9021–9043, 2008.

[Stock, 2006] Mark Stock. *A Regularized Inviscid Vortex Sheet Method for Three Dimensional Flows With Density Interfaces*. PhD thesis, University of Michigan, 2006.

[Thuerey et al., 2006] N. Thuerey, K. Iglberger, and U. Rüde. Free Surface Flows with Moving and Deforming Objects for LBM. *Proceedings of Vision, Modeling and Visualization 2006*, pages 193–200, Nov 2006.

[Thuerey et al., 2010] Nils Thuerey, Chris Wojtan, Markus Gross, and Greg Turk. A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph.*, 29(4):48:1–48:10, July 2010.

[Treuille et al., 2006] Adrien Treuille, Andrew Lewis, and Zoran Popovic. Model reduction for real-time fluids. In *Proceedings of ACM SIGGRAPH*, 2006.

[Tryggvason and Aref, 1983] G. Tryggvason and H. Aref. Numerical experiments on hele-shaw flow with a sharp interface. *J. Fluid Mech.*, 136:1–30, 1983.

[von Funck et al., 2008] Wolfram von Funck, Tino Weinkauf, Holger Theisel, and Hans-Peter Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions in Visualization and CG*, 14(6):1396–1403, 2008.

[Wang, 2004] Qian Xi Wang. Variable order revised binary treecode. *J. Comput. Phys.*, 200:192–210, 2004.

[Weissmann and Pinkall, 2010] S. Weissmann and U. Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics*, 29(4), 2010.

[Wicke et al., 2009] Martin Wicke, Matthew Stanton, and Adrien Treuille. Modular Bases for Fluid Dynamics. *ACM SIGGRAPH Papers*, 28:Article 39, Aug 2009.

[Wilcox, 1993] D. C. Wilcox. *Turbulence modelling for CFD*. DCW Industries, 1993.

[Winckelmans et al., 1996] G. S. Winckelmans, J. K. Salmon, M. S. Warren, A. Leonard, and B. Jodoin. Application of fast parallel and sequential tree coeds to computing three-dimensional flows with the vortex element and boundary element methods. *ESAIM Proc.*, 1:225–240, 1996.

[Wojtan et al., 2010] Chris Wojtan, Nils Thuerey, Markus Gross, and Greg Turk. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics*, 29,3:8, July 2010.

[Yoon et al., 2009] J.-C. Yoon, H. R. Kam, J.-M. Hong, S.-J. Kang, and C.-H. Kim. Procedural synthesis using vortex particle method for fluid simulation. *Compu. Graph. Forum*, 28(7):1853–1859, 2009.

[Yu et al., 2009] Qizhi Yu, Fabrice Neyret, Éric Bruneton, and Nicolas Holzschuch. Scalable real-time animation of rivers. *Comput. Graph. Forum*, 28(2):239–248, 2009.

[Zhao et al., 2010] Ye Zhao, Zhi Yuan, and Fan Chen. Enhancing fluid animation with adaptive, controllable and intermittent turbulence. *ACM Eurographics*, 2010.

[Zhu and Bridson, 2005] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *Proceedings of ACM SIGGRAPH*, 24(3):965–972, 2005.

Curriculum Vitae

Tobias Pfaff

Personal Data

3. Oct 1980	Born in Bühl, Germany
Nationality	German

Education

12. Jul 2012	Ph.D. defense
Mar 2008 – Jun 2012	Research assistant and Ph. D. student, ETH Zurich Advisor: Prof. Markus Gross
Sep 2007	Diploma degree in Physics *Numerical modeling and joint inversion of ERT and solute transport* Advisor: Prof. Kurt Roth, University of Heidelberg
Oct. 2001 – Sep. 2007	Studies of Physics, Universität Konstanz, Germany Specialization: Quantum Physics, Semiconductor Physics
Oct. 2003 – Apr 2006	Studies of Computer Sciene, Fernuniversität Hagen, Germany In parallel to studies of Physics Graduation with Vordiplom in Computer Science

Awards

2001 – 2007	Holder of a full scholarship, Studienstiftung des deutsches Volkes
2003 – 2004	JASSO fellowship, Japanese Education Department
2001	National winner of the 19th Bundeswettbewerb Informatik (German National Computer Science Competition)

Scientific Publications

T.PFAFF, N. THUEREY and M. GROSS. Lagrangian Vortex Sheets for Animating Fluids. In *Proceedings of ACM SIGGRAPH (Los Angeles, USA, August 5-9, 2012), ACM Transaction on Graphics, vol. 31, no.4, pp.112:1–112:8.*

T. PFAFF, N. THUEREY, J. COHEN, S. TARIQ and M. GROSS. Scalable Fluid Simulation using Anisotropic Turbulence Particles. In *Proceedings of ACM SIGGRAPH Asia (Seoul, Korea, December 15-18, 2010), ACM Transaction on Graphics, vol. 29, no. 5, pp. 174:1–174:8.*

T. PFAFF, N. THUEREY, A. SELLE and M. GROSS. Synthetic Turbulence using Artificial Boundary Layers. In *Proceedings of ACM SIGGRAPH Asia (Yokohama, Japan, 16-19, 2009), ACM Transactions on Graphics, vol. 28, no.5, pp. 121:1–121:10.*

U. WOLLSCHLÄGER, T. PFAFF and K. ROTH. Field-scale apparent hydraulic parameterization obtained from TDR time series and inverse modeling. In *Hydology and Earth System Sciences, vol. 13, pp. 1953, 2009.*

Employment and Research

Mar 2008 – Jun 2012	Research assistant, Computer Graphics Lab, ETH Zurich, Switzerland. *Turbulence methods, Fluid Simulation*
Jul 2007 – Sep 2008	Visiting researcher, Soil Physics Group, CAS Lanzhou, China. *Evaluation of ERT measurements for studying permafrost soils*
Sep 2006 – Sep 2007	Research assistant, Institute of Environmental Physics, Universität Heidelberg, Germany. *Inverse modeling of geophysical data, Solute transport models*
Feb 2005 – Apr 2005	Research internship, Bosch GmbH, Germany. *Development of algorithms for a novel 3D-optical sensor system*
Jul 2005 – Sep 2005	Visiting researcher, AI Media Lab, KAIST, Korea. *Optimal trajectory planning for bipedal robots using a full-friction model*
Mar 2002 – Sep 2002	Undergraduate research assistant, Solid-state Physics Group, Universität Konstanz, Germany. *Characterization of a novel acoustic microscope (SNAM)*