Linköping Studies in Science and Technology
Dissertation No. 1345

Memory Efficient Methods for Eulerian Free Surface Fluid Animation

Andreas Söderström

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Science and Technology
Department of Management and Engineering
Linköping University, SE-581 83 Linköping, Sweden

Linköping, October 2010

Cover, front page: A fluid simulation of water flooding a corridor. In order to capture the thin sheet of fluid near the upper middle part of the image dual resolution level set surface tracking was employed. The memory-efficient fluid animation system presented in this thesis allows this scene to be computed using less than 30MB of main memory.

Cover, back page: A fluid simulation of a fountain. This simulation was created using the out-of-core and compression framework for particle level sets presented in this thesis.

Memory Efficient Methods for Eulerian Free Surface Fluid Animation

Linköping Studies in Science and Technology
Dissertation No. 1345

Department of Science and Technology
Department of Management and Engineering
Linköping University, SE-581 83 Linköping, Sweden

ISBN: 978-91-7393-295-0 ISSN: 0345-7524

Printed by LiU-Tryck, Linköping, 2010

Abstract

This thesis focuses on improving and extending the available toolset for Eulerian, i.e. grid based, free surface fluid animation and level set based surface tracking in the context of computer graphics and visual effects. More specifically three novel methods are presented each aimed towards reducing the amount of computer memory required for producing high resolution animations of incompressible free surface fluids. Each method is primarily developed for, but not limited to, the popular Stable Fluids method (Stam, 1999).

Eulerian free surface fluid animation has historically required a large amount of computer memory, especially when high resolution results are desired. This problem has recently been addressed through the development of dynamic computational grids like the Dynamic Tubular Grid (DT-Grid) for level set computations. However, when animating free surface fluids a large amount of tracker particles are often added to the level set geometry in order to provide more accurate tracking of fluid surfaces. As a result the particle level set (PLS) method typically requires two orders of magnitude more memory than a DT-Grid level set. In order to reduce the gap in memory requirement between the level set and the particles this thesis introduces a fast and efficient compression method for such tracker particles. This compression is optionally combined with a specialized external memory algorithm that allows particle and level set data to be efficiently streamed back and forth between primary memory and secondary storage devices such as hard disk drives. The particle compression scheme is able to reduce the size of a DT-Grid particle level set by more than 65% while only inducing a 5% penalty to performance. If combined with the external memory algorithm particle level sets of virtually any size and resolution can be used in free surface fluid animations. The induced performance penalty of the combined scheme depends on the performance of the external storage device. When using a traditional hard disk drive a 70% increase in simulation time compared to the uncompressed in-core simulation was measured in the worst case.

This thesis also presents a purely Eulerian alternative to the PLS method through the introduction of a dual resolution level set representation. The method replaces the tracker particles with a level set of higher resolution, thus significantly increasing surface tracking accuracy compared to the unaided level set. The scheme is able to produce high quality results using up to 94% less memory than a PLS. The core component of the method is the Spatially Adaptive Morphology (SAM) filter which connects the high resolution representation of the level set with the lower resolution fluid, thus providing plausable animation also for small and/or thin surface features.

A sheet preserving extension to the SAM filter is also presented that is able to preserve thin sheets of fluid indefinitely if so desired. Although this method adds mass to the simulation it is highly useful for animating phenomena like splashes, fountains and waterfalls.

The final method presented in this thesis concerns the efficient local animation of oceans and other very large free surface fluids.For such scenarios large amounts of memory and computation time can be saved by only computing accurate fluid physics in a local fluid region immediately surrounding a point of interest. The fluid outside this region can then be animated using less accurate but significantly faster and less memory demanding models. However, for this approach to be accurate the local fluid must be contained in such a way that it behaves as if still part of a larger fluid. This thesis enables the local simulation of a larger body of fluid by introducing three different non-reflective boundary conditions for free surface fluid animation using a modified Stable Fluids method. Two simple wave dampening boundaries are presented as well as a significantly more advanced wave absorbing boundary based on the Perfectly Matched Layer (PML) approach. All three boundaries are shown to be effective in preventing wave reflection given large enough boundary regions. However the PML boundary is significantly more efficient, typically absorbing waves at a fraction of the distance required by the other two methods.

Preface

The story of this thesis starts with a very fortunate encounter: During the final year of my M.Sc. in applied physics I took an interest in scientific visualization. During this time I met Ken Museth, the enthusiastic and inspiring professor that would later become my supervisor and guide for the course of my PhD. During the fall of 2004 Ken asked me if I wanted to do my diploma work for him studying free surface fluid animation using level sets. Although I had no previous experience with either computational fluid dynamics or the level set method, the challenge and the opportunity to learn compelled me. What followed was six intensive months during which time I came to appreciate the strange blend of math, physics and computer science that is fluid animation. I graduated in 2005 and before the end of the year I was a PhD student in the computer graphics research group that Ken had created at Linköping University.

As a PhD student my research quickly focused on Eulerian methods for the animation of incompressible free surface fluids. At the time I joined the Linköping University graphics group Michael B. Nielsen was working with professor Ken on a highly memory efficient computational grid for level set animation. I was given the exciting challenge to develop a free surface fluid animation framework based on these Dynamic Tubular Grids (DT-Grids).In parallel to this work I also started teaching part-time at Linköping University.

In the spring of 2006 I joined professor Museth on a two month visit to Rhythm and Hues Studios in Los Angeles. This trip was a great source of inspiration as well as a valued opportunity to experience visual effects research in a production environment.

In fall 2006 the DT-Grid based fluid animation system was finally completed and the result was, to the extent of our knowledge, the most memory efficient system for Eulerian free surface fluid animation (using uniform sampling) to date. At this time I had also started working with professor Museth and PhD students Michael B. Nielsen and Ola Nilsson on a specialized framework for compression and out-of-core streaming of level sets and particle level sets. My fluid solver was included in the resulting system and our combined work was presented first as an SIGGRAPH sketch [Nielsen et al., 2006] and later published in Transactions on Graphics (TOG) as Paper I [Nielsen et al., 2007].

In parallel to the development of the fluid animation system I had also worked with PhD students Gunnar Johansson (now Gunnar Läthén) and Ola Nilsson on building a cheap open-source rendering cluster. The purpose of this cluster was to render the high resolution DT-Grid level sets we were now able to produce. Our work was presented as a work-in-progress at

SIGGRAD 2006 [Johansson et al., 2006] and completed later that year.

During 2007 I primarily spent my time taking courses and teaching. I was also experimenting with the use of DT-Grids for engineering problems, specifically finding the equilibrium profile of a sessile drop under Van-Der-Waal stress. However, although we achieved fairly accurate results our method could not compete with explicit alternatives. In the spring of 2007 Ola Nilsson, Gunnar Läthén and I was also given responsibility for the course "Modeling and Animation" filling in for professor Museth who was on a leave of absence. We decided to complement the solid theoretical foundation of this course with a strong practical component. Consequently we spent a significant effort in developing a completely new lab series ranging from mesh data structures and decimation to level sets and fluid animation. Our efforts were rewarded when the students at the end of the course rated it as outstanding, awarding us with an acknowledgment from the Dean. I have now been involved in teaching this popular course for several years.

In the spring of 2008 I interned at the visual effects company Digital Domain, once again working with professor Museth on visual effects in a production environment. This work was a highly rewarding and inspirational experience from both a personal and professional perspective. Upon my return I started research into two new projects: An Eulerian alternative to the particle level set method and non-reflective boundaries for free surface fluid animation. My effort to be rid of the level set particles later resulted in Paper II [Söderström and Museth, 2010], and my work on non-reflective boundaries would lead to Paper III [Söderström et al., 2010]. I also started researching adaptive grids for Eulerian fluid animation and DT-Grids. Furthermore Paper I was invited to be presented at the SIGGRAPH 2008 conference.

During the fall of 2008 professor Ken Museth decided to leave Linköping University and join Digital Domain. To my relief and appreciation Ken still agreed to stay on as adjunct professor and as my supervisor for the remainder of my PhD. However, with my supervisor now living in California I decided to approach professor Matts Karlsson at Linköping University and ask him to be my co-supervisor. My offer was accepted and Professor Matts and his students has been a much appreciated support during the final part of my PhD.

During 2009 I focused primarily on finalizing my current research projects. This resulted in a sketch presentation at SIGGRAPH 2009 [Söderström and Museth, 2009] as well as the presentation of Paper II at Eurographics 2010. In parallel to this I also finalized Paper III which was later accepted for publication in TOG. During this time part of my work was also presented at a local exhibition coinciding with the opening of the Norrköping Visualization Center.

During the summer and fall of 2010 my PhD was concluded with the writing of this thesis.

Acknowledgments

During the course of my PhD I have had the privilege to meet and work with a number of highly skilled and influential people. I would now like to take this opportunity to express my deepest gratitude to these and all the other people that have contributed to making this thesis possible.

First and foremost I would like to express my deepest gratitude and appreciation to my supervisor Ken Museth. Your knowledge, enthusiasm and creativity has been an constant source of inspiration for me. Without your guidance and insight I would not be where I am today. Thank you.

A special thanks also goes to my co-supervisor Matts Karlsson who provided me with a second home at the department of management and engineering during the latter half of my PhD. He has rekindled my interest in engineering and his knowledge, insight and support has been invaluable.

Next it is my pleasure to thank past and present members of professor Museth's Graphics Group. Working in this diverse group of highly skilled and intelligent people has been a stimulating and rewarding experience both personally and professionally. I would especially like to thank Michael B. Nielsen and Ola Nilsson. Part of this work is based on their efforts and research. I would also like to thank Gunnar Läthén for his work with the rendering cluster and for many rewarding conversations.

My appreciation and thanks also goes to my talented colleagues at the departments of ITN and IEI. I especially wish to thank Björn Gudmundsson and Reiner Lenz for their support and Gun-Britt Löfgren and Anna Wahlund for their invaluable administrative work.

Next I would like to extend my thanks to all the exceptional people I met during my visits to Rhythm and Hues studios and Digital Domain. I especially wish to thank Jerry Tessendorf at Rhythm and Hues and Doug Roble at Digital Domain. Jerry Tessendorf guided me through the exciting world of visual effects production at Rhythm and Hues and Doug Roble was an invaluable contact and inspiration during my time at Digital Domain. A special thanks also goes to professor Ken Museth for his integral part in making these visits possible.

I also wish to acknowledge the Stanford 3D Scanning Repository for providing several of the high resolution models used in my work.

Last but not least I wish to express my gratitude to my friends and family. Their constant support has been instrumental in finding the energy needed throughout my PhD and the writing of this thesis. A special mention goes to my close friends and proof-readers Thomas Nyberg and Emma Löfstedt for their comments, friendship and support. However, my most special thanks goes to my beloved Kristina. Without you my life would be empty and this thesis would never have been written.

List of Publications

This thesis is based on the following papers, which will be referred to in the text by their Roman numerals:

I. M. B. Nielsen, O. Nilsson, A. Söderström, and K. Museth. Out-of-core and Compressed Level Set Methods. *ACM Trans. Graph.*, 26 (4), 2007. ISSN 0730-0301

II. A. Söderström and K. Museth. A Spatially Adaptive Morphological Filter for Dual-Resolution Interface Tracking of Fluids. pages 5–8, Norrköping, Sweden, 2010. Eurographics Association. ISBN undefined. URL `http://www.eg.org/EG/DL/conf/EG2010/short/005-008.pdf`

III. A. Söderström, M. Karlsson, and K. Museth. A PML Based Non-Reflective Boundary for Free Surface Fluid Animation. *ACM Trans. Graph.*, 29(5), 2010. doi: 10.1145/1857907.1857912

(articles reprinted with permission)

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
CG	Computer Graphics
CPT	Closest Point Transform
DB-Grid	Dynamic Block Grid
DT-Grid	Dynamic Tubular Grid
DV-Grid	Dynamic Volume Grid
GB	Gigabyte (1 000 000 000 bytes)
HDD	Hard Disk Drive
I/O	Input / Output
kB	Kilobyte (1 000 bytes)
LB	Liquid Biased (filter)
LRU	Least Recently Used
LSM	Level Set Method
MB	Megabyte (1 000 000 bytes)
MLS	Marker Level Set
ODE	Ordinary Differential Equation
OOC	Out-Of-Core
PDE	Partial Differential Equation
PLS	Particle Level Set
PML	Perfectly Matched Layer
RAID	Redundant Array of Independent Disks
SAM	Spatially Adaptive Morphology (filter)
SP-SAM	Sheet Preserving Spatially Adaptive Morphology (filter)
SSD	Solid State Drive
VFX	Visual effects

Contents

Abstract i

Preface iii

Acknowledgments v

List of publications vii

Abbreviations ix

Table of Content xii

1 Introduction 1
- 1.1 Fluid Animation for Visual Effects - a Brief History 1
- 1.2 Commercial Packages and Implementations 4
- 1.3 Aims and Motivation 4
- 1.4 Originality and Contributions 5
- 1.5 Outline of the Thesis 6

2 Eulerian Free Surface Fluid Animation 9
- 2.1 The Navier-Stokes Equations 9
- 2.2 Simplified Fluid Models 15
- 2.3 Boundary Conditions 17
- 2.4 Surface Tracking 20
- 2.5 Memory Efficient Eulerian Simulation 28
- 2.6 Constructing an Eulerian Fluid Solver 33
- 2.7 Artificial Conservation of Volume and Energy 40

3 Breaking the Memory Barrier - Out-of-Core and Compressed Level Set Methods 47
- 3.1 Introduction 47
- 3.2 Background 48
- 3.3 Compression for DT-Grid Based Free Surface Fluid Animation 48
- 3.4 Efficient Out-of-Core Streaming for DT-Grid Based Fluids . 56
- 3.5 Summary 64

4 Preserving Surface Details Without the Particle Pain 67
- 4.1 Introduction 67
- 4.2 Related Work 68
- 4.3 The Spatially Adaptive Morphology Filter 69

4.4 The Sheet Preserving SAM Filter 70
4.5 Results 73
4.6 Summary 80

5 An Ocean in a Box - Non-Reflecting Boundaries for Free Surface Fluid Animation 83
5.1 Introduction 83
5.2 Background 85
5.3 Constructing Dampening Regions 86
5.4 Measuring Wave Absorption Efficiency 88
5.5 Explicit Dampening 91
5.6 Implicit Dampening 93
5.7 Wave Absorbing Boundaries - The Perfectly Matched Layer Approach 96
5.8 PML Boundaries on MAC Grids 118
5.9 Summary 119

6 Conclusions 121
6.1 The Out-Of-Core and Compression Framework (Paper I) . 121
6.2 The Dual Resolution Surface Tracking Method (Paper II) . 123
6.3 The Non-Reflecting Boundaries (Paper III) 125
6.4 Concluding Remarks 128

Bibliography 128

Chapter 1

Introduction

In this chapter background material for this thesis is provided in the form of a brief history of fluid animation for visual effects. A number of prominent commercial tools for fluid animation are also presented as well as several open-source alternatives. After this the novelties and contributions of the thesis are briefly discussed followed by an outline of the remainder of the thesis.

1.1 Fluid Animation for Visual Effects - a Brief History

The last couple of decades have seen a remarkable change in the way motion pictures are produced. The advent of the digital computer has allowed artists and film makers to bring to life their visions and dreams in ways few believed possible only thirty years ago. We have witnessed how computer generated (CG) visual effects have progressed from the simple X-wing targeting computers in the original *Star Wars* motion picture to the breathtaking CG world brought to life in the movie *Avatar*. However, as more and more CG scenery makes its way into motion pictures the necessity for accurate animation and physics affecting this content also increases. Few would be convinced by a ship that does not generate waves in the water or trees that do not move in the wind. A common approach over the years has been to animate CG content by hand[1]. Although there are many cases where this approach has been successful, such animation typically require significant artistic talent in order to achieve visually pleasing results. This is especially true for photorealistic CG content that is intended to be part of actual live action scenery. The context of realism provided by this scenario can make even slight mistakes in animation look unrealistic, breaking the illusion that the CG content is an actual part of the scene.

Although most types of animation can be done manually or by recording natural motion through the use of motion capture equipment, realistic animation of CG content remains a prohibitively hard task for many scenarios. One such scenario is the animation of fluid phenomena such as smoke, fire and water. The highly complex dynamics of fluids makes visually pleasing fluid animation a very hard task to perform by hand. As a result, tools

[1]Using for example simple rigging with a skeleton and motion paths.

that are able to aid in creating realistic animations of fluid phenomena are highly desirable. The construction of such a tool is however no simple task. The physics of fluids is very complex, including vortex formations and deformations, turbulence, surface tension effects, interface dynamics and more. The behavior of turbulent flow is indeed so complex it is considered one of the remaining unsolved problems in physics [Clay Mathematics Institute, 2010]. Even one of the Millennium Prize problems, i.e. the Navier-Stokes existence and smoothness problem [Fefferman, 2010], is directly related to the complexities of fluid flow.

Over the ages many renowned philosophers and thinkers have worked towards understanding and explaining the behavior of fluids. These studies stretch at least as far back as ancient Greece where Archimedes began to study static fluids and floating bodies. During the renaissance Leonardo Da Vinic studied the physics of fluids extensively through observation and his legacy include a significant collection of drawings and illustrations. However, the development of a mathematical theory for describing the behavior of fluids did not begin until the late 17:th century with the work of Sir Isaak Newton. Along with his famous second law of motion ($F = m \cdot a$) Newton also discovered, among many other things, that the stress on a fluid often relates almost linearly to strain (i.e. stress induced deformation). Many everyday fluids ,like water and air[2], exhibit this behavior and are for this reason often referred to as Newtonian fluids.

The mathematical groundwork for what can be considered modern fluid dynamics was laid down in the late 17:th and early 18:th century starting with the work of Daniel Bernoulli and Leonhard Euler. Bernoulli derived the Bernoulli equation which can for example be used for estimating fluid flow through pipe systems and Euler derived the Euler equations which provide a convincing model for an inviscid fluid. In the 18:th and 19:th century Claude-Louis Navier and George Stokes contributed the famous Navier-Stokes equations describing the dynamics of a viscous fluid.

The Navier-Stokes equations provide a very convincing model for the dynamics of fluid flow and consequently these equations have since been used extensively in the fields of science and engineering. Examples range from meteorology and the calculation of weather forecasts to ballistics and space flight. In the field of computer graphics and visual effects the Navier-Stokes equations have become the backbone for a range of computer animated fluid effects, including smoke, fire, wind and water.

The primary focus of fluid animation for visual effects is however not, as one might believe, accurate simulation of the physics of fluids. The primary concern is to provide *visually pleasing animation* of fluids, which may or may not include physical accuracy. In essence “what looks good is good”. As a result a special breed of methods have been developed in the field of visual effects for solving the Navier-Stokes equations. These methods are

[2]From a physics perspective both liquids and gases are fluids.

designed to produce visually pleasing fluid animations while emphasizing robustness, speed and versatility. The requirements of robustness and speed stem from the fact that within the graphics industry there is often a very limited timeframe for producing a fluid animation. For a game fluid animations must be possible to compute in real-time whereas for motion pictures a decent fluid animation should often be possible to compute overnight. With these relatively tight time constraints it is imperative that the methods employed for solving the Navier-Stokes equations are able to produce physically plausible and visually pleasing solutions at all times with no exceptions. Versatility is also an important factor since visual effects involve a significant artistic component. Thus a fluid animation system for visual effects should be as general-purpose as possible, allowing a plausible fluid animation to be computed regardless of any far-fetched or even unphysical scenarios created by the animator.

One of the first methods in computer graphics for solving the Navier-Stokes equations that met all these requirements is the celebrated *Stable Fluids* method first presented by Jos Stam [Stam, 1999], building upon important developments by among others Nick Foster and Dimitri Metaxas [Foster and Metaxas, 1996, 1997a,b]. The Stable Fluids method provides an *unconditionally stable* approach to animating fluids and quickly became a cornerstone for the animation of many fluid phenomena including smoke [Fedkiw et al., 2001; Stam, 1999], fire [Nguyen et al., 2002] and water [Foster and Fedkiw, 2001]. Since 1999 and the Stable Fluids method, research into the field of fluid animation for visual effects has exploded with numerous improvements and developments to the original Stable Fluids method. Some examples include the application of vorticity confinement [Steinhoff and Underhill, 1994], vortex particles [Selle et al., 2005], improved integration schemes [Dupont and Liu, 2003; Molemaker et al., 2008; Mullen et al., 2009; Yabe et al., 2001] more accurate boundaries [Batty et al., 2007] and more.

Alternative methods for solving the Navier-Stokes equations have also been explored. Noteworthy examples include the Smoothed Particle Hydrodynamics (SPH) method [Ellero et al., 2007; Monaghan, 1988], the Fluid Implicit Particle (FLIP) approach [Brackbill et al., 1988; Zhu and Bridson, 2005], Lattice Boltzmann solvers [Chen et al., 1992; Fan et al., 2005] and fluids on tetrahedral meshes [Batty et al., 2010; Chentanez et al., 2007; Klingner et al., 2006]. A good overview of some of the recent developments of fluid simulation in the field of computer graphics can be found in the book “Fluid Simulation for Computer Graphics” [Bridson, 2008] as well as the recent survey by Tan and Yang [2009].

Currently the Stable Fluids method and the subsequent developments in the field of fluid animation for visual effects have led to fluid animations playing an important role in many motion pictures in recent years. Examples include blockbusters such as *The Lord of The Rings*, *The Day after Tomorrow*, *Pirates of the Caribbean*, *2012* and *Avatar* among many others.

1.2 Commercial Packages and Implementations

There are currently many commercial visual effects tools and packages that support fluid effects. Prominent examples include software packages such as Autodesk *Maya* and *Houdini* by Side Effects Software. In addition to general-purpose tools such as these there are a number of commercial packages that specialize on fluid effects in particular. Two such tools are *Realflow* by Next Limit Technologies and *Flowline* by Scanline VFX. The recent upstart *Naiad* by Exotic Matter has also recently made a name for itself with its use in the motion picture *Avatar*.

In addition to commercial packages there are also a number of open source tools available for producing fluid effects. The open source 3D suite *Blender*[3] contains a Lattice Boltzmann fluid solver and *iSPH*[4] is an open source SPH solver. The large *OpenFOAM*[5] package also contains numerous fluid simulation tools; however it is primarily aimed towards science and engineering applications.

In spite of this offering of fluid animation tools it is common for large visual effects studios to maintain their own in-house fluid animation system. This can be advantagous since a studio has full control over its in-house tools, allowing for specialized tweaks and modifications in order to customize effects for a particular production or scene. However, maintaining a state-of-the-art fluid animation tool is a costly process and as commercially available software is getting more and more advanced the advantages of this approach will likely become smaller.

1.3 Aims and Motivation

In 2005 when we[6] started research into fluid animation for visual effects many of the current methods were, and arguably still are, lacking in a number of areas. One area where we saw great potential for improvement was the Eulerian, i.e. grid-based, simulation of free surface fluids using level set [Osher and Fedkiw, 2002] surface tracking. Consequently the overall aim of this thesis became the improvement and extension of the available toolset for Eulerian free surface fluid animation and level set based surface tracking.

Since high definition Eulerian level set and free surface fluid animations were found to require a sometimes prohibitively large amount of computer memory this problem quickly became the main focus of this thesis. Our research initially led to the development of the Eulerian free surface fluid animation system based on Dynamic Tubular Grids (DT-Grids) described

[3]http://www.blender.org/
[4]http://isph.sourceforge.net/
[5]http://www.openfoam.com/
[6]The Linköping University Graphics Group, http://gg.itn.liu.se/

briefly in section 2.6.2. However, although memory efficient, this system could still be improved further leading to the external memory and compression methods presented in Paper I, the surface tracking method of Paper II and finally the boundary conditions presented in Paper III.

1.4 Originality and Contributions

The papers included in this thesis represent a number of original ideas and novel contributions to the fields of computer graphics and Eulerian free surface fluid animation. Behind each paper is also a large amount of practical work in order to implement, test and verify said ideas. Of the papers included in this thesis I am the primary author of Paper II and Paper III. In Paper I I worked as part of a team contributing both ideas and implementations with my primary focus being the fluid animation aspects of this paper. The ideas presented in Paper II and Paper III as well as the implementations necessary for demonstrating their effectiveness are to a large extent my own. A brief summary of my contributions to the respective papers included in this thesis is provided below:

Paper I For this paper the main focus of my work and contributions are the fluid animation aspects of out-of-core and compressed level set simulations. I have contributed the streaming and compression framework for the particle level set method[Enright et al., 2002a] as well as the out-of-core fluid animation system that is a part of the paper. I have also played a supporting role in the development of the streaming and compression schemes for regular level sets presented in this paper.

Paper II The ideas and execution behind this paper is to a large extent my own work. I have contributed the ideas behind the SAM and SP-SAM filters as well as their use in presenting a memory efficient Eulerian alternative to the particle level set method. I have also contributed all the practical work in order to test and evaluate the resulting method.

Paper III This paper constitutes to a large extent my own ideas and execution. I have contributed all the central ideas including the Stable Fluids based solution algorithm and the related stabilization scheme for the PML equations for incompressible free surface flow. The analysis of the performance of said algorithm as well as the arbitrary level set boundaries presented is also my own work.

Related to the work presented in this thesis is also the following minor publications and technical reports to which I have contributed:

[Johansson et al., 2006] : G. Johansson, O. Nilsson, A. Söderström, and K. Museth. Distributed ray tracing in an open source environment (work in progress). pages 7–11. Linköping University Electronic Press, 2006

[Nilsson and Söderström, 2007] : O. Nilsson and A. Söderström. Euclidian distance transform algorithms : A comparative study. Technical report, Linköping UniversityLinköping University, Department of Science and Technology, 2007

[Nielsen et al., 2006] : M. B. Nielsen, O. Nilsson, A. Söderström, and K. Museth. Virtually infinite resolution deformable surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 66, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6

[Söderström and Museth, 2009] : A. Söderström and K. Museth. Non-reflective boundary conditions for incompressible free surface fluids. In *SIGGRAPH '09: SIGGRAPH 2009: Talks*, pages 1–1, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-834-6

1.5 Outline of the Thesis

The remainder of this thesis is divided into five chapters, the contents of which is outlined below:

Chapter 2 introduces the reader to the theory and methodology behind fluid animation in general and memory efficient animation of incompressible Eulerian free surface fluids in particular. This chapter introduces many of the concepts, methods and data structures upon which the contributions of this thesis are built. The intended reader is not an expert in Eulerian fluid animation and thus readers that are very familiar with the Navier-Stokes equations, the level set method and sparse computational grids may safely skip most of this chapter. However, it is still recommended to read section 2.6.2 which introduces the fluid animation system used for all numerical experiments presented throughout this thesis and its related papers.

Chapter 3 focuses on the fluid animation aspects of the out-of-core and compressed level set and particle level set framework presented in Paper I. The chapter discusses the core methods and results and can be seen as complementary reading to Paper I. The chapter is concluded with short summary discussing the strengths and weaknesses of the framework.

Chapter 4 describes the Eulerian method for dual-resolution tracking of free surface presented in Paper II. The chapter provides background information for this project as well as the core method and results. This chapter is intended as complimentary reading to Paper II. The chapter is concluded with a short summary discussing the strengths and weaknesses of the method.

Chapter 5 concerns the non-reflective boundaries for incompressible free surface fluid animation presented in Paper III. The chapter puts more emphasis on the explicit and implicit dampening methods presented in this paper and also provides a more extensive version of the derivation of the PML boundary equations. The chapter also shows how to derive PML boundary equations for arbitrary external forces. This chapter is intended as complimentary reading to Paper III. The chapter is concluded with a short summary discussing the strengths and weaknesses of the method.

Chapter 6 concludes this thesis with a discussion on the overall contributions of the thesis, its potential impact and ideas for future work.

After these chapters follows an appendix containing the core papers of the thesis.

Chapter 2

Eulerian Free Surface Fluid Animation

All the contributions in this thesis concern the animation of free surface fluids in the context of computer graphics and visual effects. More specifically this thesis focuses on Eulerian (i.e. grid-based) animation of incompressible free surface fluids. As part of the research behind Paper I, Paper II and Paper III a framework for computing such visual effects through the simulation of incompressible free surface fluids was developed. This chapter provides an introduction to the theory and methods employed by this framework as well as a quick overview of a number of related methods.

The Navier-Stokes equations are at the core of many fluid animation systems, including the system constructed as part of this thesis. Section 2.1 presents how these equations can be obtained from Newton's second law of motion together with conservation laws for mass and energy. Section 2.1 also describes the Lagrangian and Eulerian viewpoints commonly associated with the Navier-Stokes equations.

Section 2.2 proceeds by presenting a number of simplifications to the Navier-Stokes equations that are commonly used for fluid animation. In section 2.3 several common boundary conditions for the incompressible Navier-Stokes equations are then presented including the free surface boundary used for free surface fluid animation. Next section 2.4 introduces several methods for representing and tracking free surfaces, primarily focusing on the Eulerian methods used throughout this thesis. Section 2.5 continues by presenting a number of methods for efficiently storing the computational grids required for Eulerian fluid animation. This section also introduces the memory efficient Dynamic Tubular Grid (DT-Grid) that forms the foundation for many of the contributions presented in this thesis. Next section 2.6 presents the Stable Fluids based method used as a basis for the animation of incompressible free surface fluids in this thesis. This chapter is concluded with section 2.7 presenting two simple methods for avoiding undesired loss of mass and energy in Stable Fluids based animations.

2.1 The Navier-Stokes Equations

Most fluid animation systems for visual effects are based on the model for fluid flow provided by the Navier-Stokes equations. Consequently these equations form the very foundation of this thesis and the research it presents. The Navier-Stokes equations can be derived from three fundamental physical principles; the conservation of linear momentum, the conservation of mass and the conservation of energy. In this section we will show how the Navier-Stokes momentum equations can be obtained from Newton's second law of motion describing the conservation of (linear) momentum. The equation for the conservation of mass will also be presented. The equation related to the conservation of energy will be presented separately in section 2.1.2.

A formal derivation of the Navier-Stokes equations is a lengthy process and we refer the interested reader to relevant textbooks such as [Anderson, 1995; Klarbring, 2006]. However, fundamental understanding of these equations and the physics contained within them can still be provided through the relatively informal derivation of these equations as presented in this section:

We start by considering Newton's second law of motion for a single point mass (i.e. particle), commonly written as

$$\frac{d}{dt}\left(m_p \mathbf{v}\right) = \mathbf{f} \tag{2.1}$$

Here $\mathbf{v} = \{v_x, v_y, v_z\}$ describes the particle velocity, m_p represents its mass and $\mathbf{f}$ represents external forces. Equation (2.1) is well suited for modeling the dynamics of solid objects since such motion can readily be described as translations and rotations affecting the mass center of an object, i.e. translations and rotations around a single point. However, the flowing nature of a fluid makes it ill suited to be modeled as a single point. Instead fluids can be seen as a continuus and dynamic mass distribution spanning an entire region. Let Ω define this three-dimensional, continuous and finite region of fluid and let $\partial\Omega$ define the boundary of Ω. In order to animate a fluid as opposed to a single point mass we need to find the equivalent of equation (2.1) for the fluid region Ω.

Here we assume that instead of a single point mass, Ω consists of n particles. Each particle carrying a potentially different velocity $\mathbf{v}_p$, a finite mass m_p and a volume V_p . Given that Ω has a constant total mass m we observe that

$$\sum_n m_p = m \tag{2.2}$$

We now assume that each particle in Ω is still governed by equation (2.1). Since we have a cloud of particles the force $\mathbf{f}$ acting on each particle can be further divided into internal forces $\mathbf{f}_{int}$ caused by interactions between the particles contained in Ω and external forces $\mathbf{f}_{ext}$ that are independent of the particles. These forces result in the equation of motion

$$\frac{d}{dt}\left(m_p \mathbf{v}\right) = \mathbf{f}_{int} + \mathbf{f}_{ext} \tag{2.3}$$

for each particle in Ω. We now assume that the number of particles n goes towards infinity and consequently that the fluid volume V_p represented by each particle goes towards zero while still satisfying equation (2.2). As a result we may replace the particle mass m_p in equation (2.3) with the mass density ρ such that

$$\int_{\Omega} \rho d\Omega = m \tag{2.4}$$

If we assume that all particles have the same mass and volume the density ρ can also be interpreted as representing the density of particles at a point in space. It can be shown [Anderson, 1995] that for a fluid the internal stresses can be characterized by the stress tensor $\mathbf{\Sigma}$ and the resulting internal forces $\mathbf{f}_{int}$ can be calculated through the the tensor product $\nabla \cdot \mathbf{\Sigma}$. The resulting equation becomes

$$\frac{d}{dt}(\rho \mathbf{v}) = \nabla \cdot \mathbf{\Sigma} + \mathbf{f}_{ext} \tag{2.5}$$

where

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} \tag{2.6}$$

Here σ represents normal stresses and τ represents shear stresses. The internal pressure of the fluid is often a parameter of interest and thus the stress tensor $\mathbf{\Sigma}$ is commonly split into the pressure field p and the deviatoric stress tensor[1] $\mathbf{T}$ where

$$\mathbf{\Sigma} = -p\mathbf{I} + (\mathbf{\Sigma} + p\mathbf{I}) = -p\mathbf{I} + \mathbf{T} \tag{2.7}$$

In equation (2.7) $\mathbf{I}$ is the identity tensor and

$$\mathbf{T} = \begin{pmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{pmatrix} \tag{2.8}$$

where $\sigma_{xx} + p \equiv \tau_{xx}$, $\sigma_{yy} + p \equiv \tau_{yy}$ and $\sigma_{zz} + p \equiv \tau_{zz}$. As a result equation (2.5) can now be written as

$$\frac{d}{dt}(\rho \mathbf{v}) = -\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f}_{ext} \tag{2.9}$$

Equation (2.9) is the result of the physical principle that the momentum of the fluid should be conserved. However, we also have the principle that the mass of the fluid should be conserved, i.e. equation (2.4). This equation can be rewritten into the corresponding partial differential equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{2.10}$$

[1]Also known as the stress deviator tensor.

the full derivation of which can be found in for example [Klarbring, 2006]. Equation (2.10) essentially states that for every point in Ω the amount of mass flowing into that point should equal the amount of mass flowing out of it. Although equation (2.9) and (2.10) describe the dynamics of a fluid these equations are not yet in a solvable form. In order to solve them we also need to describe how to calculate the components of the stress tensor $\mathbf{\Sigma}$. This is where we go from describing the motion of a general fluid to creating a model for a specific type of fluid. The characteristics of a fluid can essentially be described through two material parameters, typically denoted λ and μ. However, these parameters are often assumed to be related (see equation (2.12)) and as a result there is only one material parameter separating one fluid from another - viscosity. Viscosity can be considered the amount of internal friction in the fluid, i.e. how "sluggish" the fluid is. Typically fluids are divided into two types depending on the behavior of the viscosity parameter. A somewhat simplified[2] distinction is that a *Newtonian fluid* has a viscosity that is independent of the forces acting upon the fluid whereas this is not the case for a *non-Newtonian fluid*.

When animating fluids for visual effects the Newtonian viscosity model is typically used. Since many common fluids like air and water can accurately be described by such a model this simplification is usually sufficient. Assuming that a fluid is Newtonian it can be characterized by the constants λ and μ [Anderson, 1995] where

$$\tau_{xx} = \lambda (\nabla \cdot \mathbf{v}) + 2\mu \frac{\partial v_x}{\partial x} \tag{2.11a}$$

$$\tau_{yy} = \lambda (\nabla \cdot \mathbf{v}) + 2\mu \frac{\partial v_y}{\partial y} \tag{2.11b}$$

$$\tau_{zz} = \lambda (\nabla \cdot \mathbf{v}) + 2\mu \frac{\partial v_z}{\partial z} \tag{2.11c}$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} \right) \tag{2.11d}$$

$$\tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \tag{2.11e}$$

$$\tau_{yx} = \tau_{zy} = \mu \left(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z} \right) \tag{2.11f}$$

As previously mentioned λ and μ are frequently assumed to be related. Typically the relation

$$\lambda = -\frac{2}{3}\mu \tag{2.12}$$

[2] A Newtonian fluid is a fluid where the stress versus strain rate curve is linear and passes through the origin.

is used although equation (2.12) has not yet conclusively been proven [Anderson, 1995]. The material constant μ is in this case the dynamic viscosity of the fluid.

2.1.1 The Lagrangian and Eulerian Views

Equation (2.9) describes the forces acting upon each infinitesimally small fluid element (i.e. particle) as it moves around with the material flow of the fluid. This special frame of reference is often referred to as the Lagrangian view. Equation (2.10) on the other hand is written in a frame of reference that is fixed in comparison to the motion of the fluid. This frame of reference is often referred to as the Eulerian view. These two views are useful since some physical parameters move around with the flow of material whereas others do not. Temperature and velocity are examples of typical Lagrangian parameters, i.e. parameters that move with the flow. Parameters like density and pressure on the other hand are typically characteristic for a stationary point in space, i.e. the Eulerian point of view, and may be considered Eulerian parameters. If we wish to make an Eulerian measurement of a Lagrangian property, like temperature, we simply apply the chain-rule

$$\frac{d}{dt}f(\mathbf{x}(\mathbf{t}),t) = \frac{\partial f(\mathbf{x},t)}{\partial t} + \nabla f(\mathbf{x},t) \cdot \frac{d\mathbf{x}(\mathbf{t})}{dt} \tag{2.13}$$

thus taking the moving coordinate frame into account. Here $\mathbf{x}(t)$ describes the motion of the coordinate frame, i.e. in this case the motion of the particles, and thus $\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}$ where $\mathbf{v}$ is the Eulerian fluid velocity. In fluid dynamics literature the notation $\frac{D}{Dt}$ is often encountered where

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla \tag{2.14}$$

is referred to as the "material derivative" and is a derivative of a property that moves with the material flow of the fluid. As can be seen from equation (2.13) this is simply a special case of the total derivative $\frac{d}{dt}$ when following the fluid flow, i.e. when $\frac{d\mathbf{x}}{dt} \equiv \mathbf{v}$. No matter the nomenclature equation (2.13) and (2.14) can both be used to obtain Eulerian measurements of Lagrangian properties. By using equation (2.13) equation (2.9) can now be written in its Eulerian form

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) + \mathbf{v} \cdot \nabla(\rho\mathbf{v}) = -\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f}_{ext} \tag{2.15}$$

Note that in this form $\mathbf{v}$ is the Eulerian fluid velocity, i.e. a velocity vector field that is fixed in space instead of moving with the flow. The Lagrangian and Eulerian views, although interchangeable in theory, makes a significant practical difference when designing an algorithm for solving the

Navier-Stokes equations. The Lagrangian view typically leads to particle based solvers like the Smoothed Particle Hydrodynamics (SPH) method [Ellero et al., 2007; Monaghan, 1988] while the Eulerian view leads to a grid-based approach like the Stable Fluids method [Stam, 1999]. Hybrid Eulerian/Lagrangian methods also exist, a good example of which is the Fluid Implicit Particle (FLIP) approach [Brackbill et al., 1988; Zhu and Bridson, 2005]. However, in this thesis the focus is primarily on Eulerian methods and thus we will use the Eulerian point of view henceforth.

2.1.2 Energy Conservation

So far we have discussed the conservation of momentum and the conservation of mass. This only leaves the conservation of energy. Using the first law of thermodynamics one can derive [Anderson, 1995] the equation

$$\begin{aligned}\frac{\partial}{\partial t}\rho\left(e+\frac{\mathbf{v}\cdot\mathbf{v}}{2}\right)+\nabla\cdot\rho\left(e+\frac{\mathbf{v}\cdot\mathbf{v}}{2}\right)\mathbf{v} &= \rho\dot{q}+\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)\\ &+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)-\frac{\partial v_x p}{\partial x}-\frac{\partial v_y p}{\partial y}-\frac{\partial v_z p}{\partial z}\\ &+\frac{\partial v_x\tau_{xx}}{\partial x}+\frac{\partial v_x\tau_{yx}}{\partial y}+\frac{\partial v_x\tau_{zx}}{\partial z}+\frac{\partial v_y\tau_{xy}}{\partial x}+\frac{\partial v_y\tau_{yy}}{\partial y}\\ &+\frac{\partial v_y\tau_{zy}}{\partial z}+\frac{\partial v_z\tau_{xz}}{\partial x}+\frac{\partial v_z\tau_{yz}}{\partial y}+\frac{\partial v_z\tau_{zz}}{\partial z}+\rho\mathbf{f}_{ext}\cdot\mathbf{v}\end{aligned}\tag{2.16}$$

describing the conservation of energy. Here e represents internal energy, κ is thermal conductivity, T is temperature and $\dot{q}$ is the local heat flux. Note that $e+\frac{\mathbf{v}\cdot\mathbf{v}}{2}$ describes the total energy - kinetic and internal. At this point we have equation (2.10) describing conservation of mass, equation (2.15) describing conservation of momentum and equation (2.16) describing conservation of energy. These equations does however not yet form a complete system since we have more unknowns than equations. In order to complete the fluid dynamics model we need to model the behavior of the "gas" of particles that makes up the fluid. Here the assumption can be made [Anderson, 1995] that

$$p=\rho RT \tag{2.17}$$

i.e. that the fluid behaves like an ideal gas where R is the specific gas constant. If the gas is also calorically perfect the equation

$$e=c_vT \tag{2.18}$$

is obtained where c_v is the specific heat at constant volume.

Equation (2.10), (2.15) and (2.16) form the Navier-Stokes equations for a compressible Newtonian fluid which together with the models in equations (2.17) and (2.18) describe the dynamics of an ideal, calorically perfect gas.

That the Navier-Stokes equations are a set of conservation laws is readily seen when the equations are written on conservation form:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_1}{\partial x} + \frac{\partial \mathbf{f}_2}{\partial y} + \frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{g} \tag{2.19}$$

where

$$\mathbf{f}_1 = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p - \tau_{xx} \\ \rho v_x v_y - \tau_{xy} \\ \rho v_x v_z - \tau_{xz} \\ \rho\left(e + \frac{\mathbf{v}\cdot\mathbf{v}}{2}\right) u_x + p u_x - k\frac{\partial T}{\partial x} - v_x\tau_{xx} - v_y\tau_{xy} - v_z\tau_{xz} \end{pmatrix} \tag{2.20a}$$

$$\mathbf{f}_2 = \begin{pmatrix} \rho v_y \\ \rho v_x v_y - \tau_{xy} \\ \rho v_y^2 + p - \tau_{yy} \\ \rho v_y v_z - \tau_{yz} \\ \rho\left(e + \frac{\mathbf{v}\cdot\mathbf{v}}{2}\right) u_y + p u_y - k\frac{\partial T}{\partial y} - v_x\tau_{yx} - v_y\tau_{yy} - v_z\tau_{yz} \end{pmatrix} \tag{2.20b}$$

$$\mathbf{f}_3 = \begin{pmatrix} \rho v_z \\ \rho v_x v_z - \tau_{xz} \\ \rho v_y v_z - \tau_{yz} \\ \rho v_z^2 + p - \tau_{zz} \\ \rho\left(e + \frac{\mathbf{v}\cdot\mathbf{v}}{2}\right) u_z + p u_z - k\frac{\partial T}{\partial z} - v_x\tau_{zx} - v_y\tau_{zy} - v_z\tau_{zz} \end{pmatrix} \tag{2.20c}$$

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ \rho\left(e + \frac{\mathbf{v}\cdot\mathbf{v}}{2}\right) \end{pmatrix} \tag{2.20d}$$

$$\mathbf{g} = \begin{pmatrix} 0 \\ f_x \\ f_y \\ f_z \\ \rho\left(v_x a_x + v_y a_y + v_z a_z + \rho\dot{q}\right) \end{pmatrix} \tag{2.20e}$$

Here $\{f_x, f_y, f_z\}$ are the components of the external force vector.

2.2 Simplified Fluid Models

The Navier-Stokes equations represent a rather extensive system of coupled, non-linear partial differential equations. Accurately solving such a system can be a massively time-consuming task even with the aid of supercomputers. This is where one of the core principles of visual effects takes over: "What looks good is good". In visual effects physical accuracy is not a primary concern, only creating effects that are good enough to make the observer believe that they are looking at actual physics. Due to this distinction fluid animation for visual effects typically does not require the level of complexity described by equation (2.19). Instead a number of simplified fluid models are often used, the most common of which will be presented below.

2.2.1 The Incompressible Navier-Stokes Equations

Although all physical fluids are compressible to some extent there are many cases where the compressibility is very low and thus it can be useful to model the fluid as incompressible instead. For the purpose of fluid animation air and most liquids can convincingly be treated as incompressible fluids.

Assuming incompressibility simplifies the Navier-Stokes equations significantly. The energy equation is no longer necessary and assuming constant viscosity the deviatoric stress tensor $\mathbf{T}$ is significantly simplified (see equation (2.23)). Furthermore, since incompressibility implies constant density the mass conservation equation (2.10) becomes a statement that the divergence of the flow should be zero, i.e. that volume should be conserved. Following the derivation in for example [Anderson, 1995] the incompressible Navier-Stokes equations for a Newtonian fluid can be written as

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = \mathbf{f}_{ext} + \mu\nabla^2\mathbf{v} - \nabla p \tag{2.21a}$$

$$\nabla\cdot\mathbf{v} = 0 \tag{2.21b}$$

where μ is the dynamic viscosity. Here the compressible momentum equation (2.15) has become equation (2.21a) and the mass conservation equation (2.10) simplifies to equation (2.21b), i.e. a statement that the there shall be no sources or sinks in the fluid velocity field. In equation (2.21a) $\mathbf{v}\cdot\nabla\mathbf{v}$ is the “self-advection term” and describes how the fluid velocity moves with the fluid flow. The term $\mu\nabla^2\mathbf{v}$ is the “viscosity term” and describes the diffusion of velocity due to viscous forces, i.e. essentially internal friction in the fluid. The terms $\mathbf{f}_{ext}$ and ∇p describes the external and pressure forces respectively.

The incompressible Navier-Stokes equations, usually written on this form, is the core component of most visual effects systems for computer aided animation of Eulerian fluids. Note that in equation (2.21a)

$$\mu\nabla^2\mathbf{v} = \mu\begin{pmatrix} \frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial y^2} \\ \frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} + \frac{\partial^2 v_y}{\partial y^2} \\ \frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} + \frac{\partial^2 v_z}{\partial y^2} \end{pmatrix} \tag{2.22}$$

implying that

$$\mathbf{T} = \mu\begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix} = \mu\nabla\mathbf{v} \tag{2.23}$$

through identification in equation (2.19).

2.2.2 The Incompressible Euler Equations

Some fluids like water and air have very low viscosity. Thus an additional simplification of the incompressible Navier-Stokes momentum equation (2.21a) is to assume zero viscosity. This leads to the incompressible Euler momentum equation commonly written as

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right) = \mathbf{f}_{ext} - \nabla p \tag{2.24}$$

and subject to (2.21b).

2.2.3 Ocean Modeling

The Euler and Navier-Stokes equations provide a fairly accurate and general-purpose description of fluid flow. However, there are many special cases where the apparent dynamics of a fluid is relatively simple. One such scenario of relevance to this thesis is the animation of ocean surfaces. Oceans represent vast bodies of water and high accuracy animation of such volumes is typically very impractical, if at all possible. However, the general behavior of ocean surfaces, especially for calm oceans, does not require the full complexity of the Navier-Stokes equations. Instead simpler fluid models can be used for computing such animations. An overview of such models is provided by Tessendorf [2004]. A number of examples can also be found in Bridson [2008]. Typical for this type of models is that the ocean is described as a two-dimensional height-field instead of a three dimensional volume. This drastically reducing the complexity of the animation. However, although height-field based models can efficiently animate large bodies of water they typically do not allow complex interactions between the water and solid objects[3]. In order to obtain the best of both worlds a potential approach is to animate the large-scale motion of the ocean surface using ocean models while creating an accurate local simulation around regions where complex interactions are expected. The non-reflective boundaries presented in chapter 5 and Paper III are intended to facilitate this type of hybrid approach.

2.3 Boundary Conditions

The Navier-Stokes equations describe the dynamics of fluids in general. They have an infinite number of solutions corresponding to the infinite variety of fluid animation scenarios that can be constructed. Obtaining a specific solution to a specific scenario requires specifying the initial and

[3]For example breaking waves around the bow of a ship traveling on the ocean.

boundary conditions for that scenario. Simply put the initial condition defines the initial state of the fluid whereas the boundary conditions define how the fluid behaves at boundaries[4], essentially determining how it is allowed to flow from this initial state. In fluid animation two types of boundary conditions are frequently encountered; Dirichlet conditions and Neumann conditions. A Dirichlet boundary condition enforces specific values to a function at boundaries while Neumann boundary conditions enforce specific values to the first derivative of this function. Below three types of boundaries of relevance to this thesis are presented together with examples of their related Dirichlet and Neumann boundary conditions.

2.3.1 Solid Boundaries

One common example of a fluid animation boundary is solid objects.An example of a Dirichlet boundary condition associated with solid boundaries is the free-slip condition

$$\mathbf{v} \cdot \mathbf{n} = \mathbf{v}_{solid} \cdot \mathbf{n} \tag{2.25}$$

In equation (2.25) $\mathbf{v}$ is the fluid velocity at the solid surface, $\mathbf{n}$ is the surface normal and $\mathbf{v}_{solid}$ is the velocity of the solid. This Dirichlet boundary condition simply states that, at the solid surface, the normal component of the fluid velocity must equal the normal component of the velocity of the solid. Although equation (2.25) can be useful for fluid animation, especially for low viscosity fluids like water, it is in fact not physically accurate for a viscous fluid. A more accurate Dirichlet boundary condition for viscous flow is the no-slip condition

$$\mathbf{v} = \mathbf{v}_{solid} \tag{2.26}$$

stating that the fluid velocity should equal the solid velocity at the solid boundary.

An example of a Neumann boundary condition at solid boundaries is the equation

$$\nabla \mathbf{v} \cdot \mathbf{n} = \mathbf{0} \tag{2.27}$$

stating that the fluid velocity at the solid surface cannot be allowed to change in the normal direction. Equation (2.27) is also sometimes written as

$$\frac{\partial \mathbf{v}}{\partial \mathbf{n}} = \mathbf{0} \tag{2.28}$$

more explicitly stating that the change of $\mathbf{v}$ along the normal $\mathbf{n}$ should be zero.

[4]For example solid walls and obstacles.

2.3.2 Two-phase Flow and the Free Surface Boundary

There are many practical scenarios where two or more fluids of different types are involved. The animation of a waterfall for example involves the two fluids water and air. In this scenario it is not only important to accurately capture the dynamics of the two fluids; it is also very important to accurately track the motion of the interface, i.e. the water surface, separating them. There are two main reasons for this. The first is that the water surface is typically the only visible part of the water animation. The internal motions of the air and water are both invisible to the naked eye. The second reason is that the water surface defines what region of the world belongs to the water and what belongs to air, thus directly influencing the solution to the Navier-Stokes equations inside these regions. There are many approaches to tracking the interfaces separating fluids. These methods range from Eulerian methods like the Level Set Method [Adalsteinsson and Sethian, 1995; Osher and Sethian, 1988; Osher and Fedkiw, 2002; Peng et al., 1999] through hybrid methods like Particle Level Sets [Enright et al., 2002a] and Marker Level Sets [Mihalef et al., 2007] to Lagrangian mesh methods including Bargteil et al. [2006]; Hirt et al. [1970]; Navti et al. [1997] among others. The strengths and weaknesses of these methods as well as relevant theory is briefly discussed in section 2.4.

When animating two-phase fluids like water for visual effects a common simplification is to only solve the the Navier-Stokes equations in the body of water whereas the "air" region is simply treated as a void that the water may fill. This "free surface" approach can be realized by applying the continuity boundary condition

$$\nabla q \cdot \mathbf{n} = 0 \tag{2.29}$$

at the fluid interface for all quantities q. Equation (2.29) simply states that all quantities on the interface shall be constant along the normal direction $\mathbf{n}$ of the water surface in the "air" region. As can be seen this is essentially a Neumann boundary condition. The free surface boundary condition is obviously approximate, however it can still provide visually pleasing animation for scenarios where the density of the animated phase is significantly higher than that of the ignored phase.

When solving the Navier-Stokes equations for free surface Eulerian flow all computations can be localized from the entire fluid domain to the region that contains the phase of interest. This typically makes free surface fluid animations significantly faster to compute than full two-phase flow. Furthermore the assumption of a free surface allows the data storage required to compute the Eulerian simulation to be localized by using dynamic computational grids such as the Dynamic Volume Grid (see section 2.6.2), a data structure closely related to the efficient Dynamic Tubular Grid (DT-Grid) by Nielsen and Museth [2006]. Indeed the memory efficient free surface fluid simulations made possible by the DV-Grid and the DT-

Grid form the foundation for this thesis in general and papers I and II in particular.

2.3.3 Non-Reflecting Boundaries

For practical reasons there are many fluid animation scenarios where it is desirable to only simulate a portion of a larger fluid domain. This local domain can for example be the immediate neighborhood around an aircraft or a patch of water on a wide ocean (see chapter 2.2.3). In such scenarios the implication is that a wide body of fluid surrounds the region where the Navier-Stokes equations are solved and that the larger body is itself never solved for. Accurate fluid dynamics can still be obtained close to the object or region of interest. However, this requires that the inflow/outflow boundary conditions linking the solved for and unsolved regions of fluid are treated with special care. This type of "open" or "far-field" boundary conditions for the compressible Navier-Stokes equations has been extensively studied in the field of computational aeroacoustics where such boundaries are used to prevent undesired reflections of pressure waves. In the field of computational aeroacoustics and computational electromagnetics a number of non-reflecting boundary conditions has been developed in order to realize accurate open boundaries. An overview of some of the current techniques is provided by Johnson [2007] and Hu [2008].

In the field of fluid animation in general and the animation of free surface incompressible fluids in particular the reflection of pressure waves is typically not a problem since such waves cannot appear in an incompressible fluid. However, an equivalent boundary condition that is able to prevent the reflection of surface waves can be highly desirable during for example local ocean simulations as presented in section 2.2.3. To the extent of our knowledge this type of boundary was first introduced to the field of graphics and free surface fluid animation by Söderström and Museth [2009] which constitutes preliminary work on the method presented in Paper III. Non-reflecting boundaries for incompressible free surface fluid animation along with the methods presented in Paper III will be considered in more detail in chapter 5.

2.4 Surface Tracking

As is presented in section 2.3.2 accurate tracking of fluid interfaces for multiphase and free surface flow is very important. Inaccurate surface tracking directly leads to an inaccurate solution to the Navier-Stokes equations which in turn can result in poor animation. Furthermore the fluid surface is often the most visible part of the fluid, and thus even small errors in the way the surface moves can lead to a fluid that does not look real. An example of a surface tracking error can be seen in figure 2.1. Here a sphere

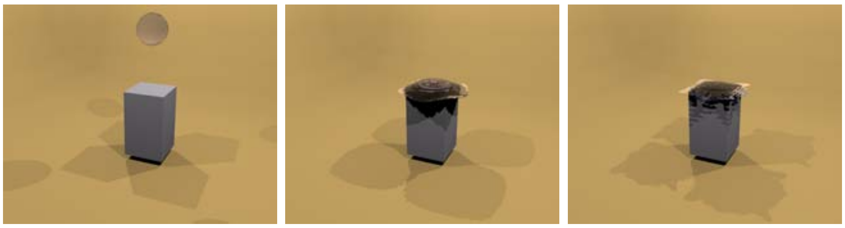


Figure 2.1: An example of failed surface tracking. Notice how the water sheets has disappeared into thin air in the image to the right.

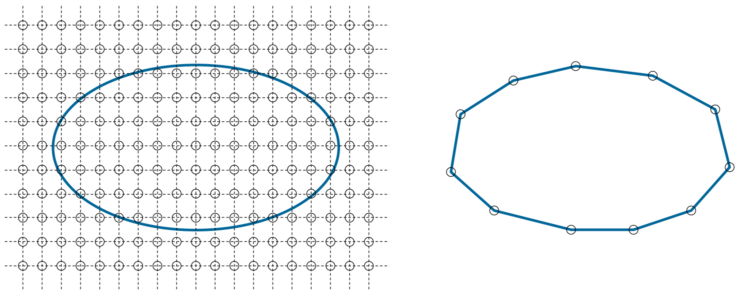


Figure 2.2: An illustration showing the difference between the Eulerian representation of an ellipsoid (left) and the Lagrangian representation (right).

of water falls onto the top of a pedestal resulting in a splash. It is hard for the human eye to determine whether the flow of water or air is physically accurate in this animation, however, it is clear that a large portion of the fluid suddenly disappears into thin air. This loss of fluid is a result of a surface tracking failure and not an error in the solution to the Navier-Stokes equations themselves.

2.4.1 Eulerian and Lagrangian Surface Tracking

As is described in section 2.1.1 there are two principally different ways to describe the motion of a fluid - the Lagrangian view and the Eulerian view. Equivalently there are two views on the surface tracking problem. The Lagrangian view corresponds to tracking surfaces explicitly by describing the motion of all points on the fluid surface. The Eulerian view corresponds to implicitly tracking surfaces by some parameter defined everywhere in space. An illustration of the difference can be seen in figure 2.2.

2.4.2 Implicit and Explicit Geometry

The Lagrangian view on geometry translates to an explicit geometric representation where every point on the geometric shape is directly defined through a set of parameters. Consider for example the equation

$$y(x) = kx + m \tag{2.30}$$

describing a line in 2D space. For any chosen value of x equation (2.30) describes a corresponding value for y such that the point (x, y) always lies on the line. Thus equation (2.30) explicitly describes where in space the line can be found. Equation (2.30) can also be written on the parametric form

$$\begin{cases} x(t) = at + x_0 \\ y(t) = bt + y_0 \end{cases} \tag{2.31}$$

effectively decoupling the variables x and y with the help of the parameter t which describes the position along the line. For any t equation (2.31) describes a point (x, y) that is on the line and thus equation (2.31) is also an explicit description of a line. Explicit geometry can typically be parameterized in this manner and thus it is often referred to as parametric geometry.

Where the Lagrangian view leads to explicit geometry the Eulerian view leads to an implicit description of geometry. Implicit geometry is as the name implies geometry that is indirectly described, typically through an equation on the form

$$f(\mathbf{x}) = 0 \tag{2.32}$$

Any point in space that satisfies equation (2.32) is considered on the implicit surface. An example of an implicit shape is the equation

$$\begin{cases} f(x, y) = x^2 + y^2 - R^2 \\ f(x, y) = 0 \end{cases} \tag{2.33}$$

which describes the implicit circle with the radius R. Equation (2.33) does not explicitly state which points (x, y) lie on the circle. However, every point in space can be tested against equation (2.33) and thus it can be determined whether or not the point is a part of the circle. In this simple case equation (2.33) can be solved analytically, thus obtaining an explicit description of the circle. However, for more complicated shapes the analytical transformation between implicit and explicit geometry can be non-trivial, if at all possible.

2.4.3 Lagrangian Surface Tracking in Practice - Meshes and Particles

Practical implementations of Lagrangian surface tracking typically correspond to a particle-based approach. A number of discrete particles are distributed in space and the particles are by definition located *on* the surface of the fluid. If topological information is also added to the particles in the form of edges and faces a mesh is formed. An example of such a mesh representation is the triangle and quad meshes commonly used for representing geometry in computer graphics. By assigning each particle a velocity based on the solution to the Navier-Stokes equations the Lagrangian surface will move in accordance to the fluid flow, thus tracking the position of the fluid surface.

There are two main advantages to this type of surface tracking. First of all particles allow for a naturally adaptive sampling of the fluid surface. Simple, low curvature regions of the fluid can be represented efficiently using few particles whereas a higher concentration can be used in complex regions. Also, the Lagrangian nature of the particles allow for accurate advection of the fluid surface. Both of these properties are very attractive for fluid animation.

Unfortunately Lagrangian surface tracking also suffer from a number of disadvantages. Fluid surfaces often exhibit complex motion with local regions undergoing severe stretching and contraction. Stretching of the fluid surface will move the tracking particles further apart, thus potentially failing to take into account any high detail fluid motion that may later occur in that region. In order to resolve this problem new particles need to be placed. This procedure requires an accurate estimation of the location of the fluid surface between points and can thus be complicated. Arguably the most significant problem with Lagrangian surface tracking is however the topology changes frequently occurring in the fluid. As two parts of the fluid collide they should merge together. This merging does however not occur naturally for explicit geometry. Instead explicit surfaces self-intersect causing situations where the surfaces become entangled and it can become non-trivial to discern how the resulting situation should be reconstructed to form a consistent merged geometry. Likewise it can be problematic to detect when and where explicit fluid surfaces should split apart.

There exists a growing body of work in the academic community on the use of explicit methods for tracking fluid surfaces. Examples include Bargteil et al. [2006]; Brochu et al. [2010]; Navti et al. [1997] among others. However, the focus of this thesis is Eulerian methods and thus the field of explicit surface tracking will not be investigated further.

2.4.4 Eulerian Surface Tracking in Practice - The Level Set Method

Surface tracking methods which rely on implicit geometry in order to track the interfaces between fluids is typically referred to as Eulerian surface tracking. A popular realization of Eulerian surface tracking is the Level Set Method (LSM) [Adalsteinsson and Sethian, 1995; Osher and Sethian, 1988; Osher and Fedkiw, 2002; Peng et al., 1999]. Level set surface tracking is a grid-based approach where the implicit function is uniformly sampled everywhere in space using a computational grid (see section 2.5). This approach may seem overly complicated and inefficient compared to Lagrangian methods. However, the development of narrow band computation [Adalsteinsson and Sethian, 1995] and compact level set representations [Houston et al., 2006; Nielsen and Museth, 2006] and Paper I has greatly reduced the computational and memory efficiency gap between explicit an implicit geometry. Even without these optimizations level set surface tracking brings with it several major advantages over explicit methods:

- Level set geometry can handle topology changes (merging and bifurcation) of arbitrary complexity in a completely robust and reliable manner.

- Surface properties like surface normals and mean curvature can readily and accurately be derived at any point in space where the level set function is defined.

- When the level set function is a signed distance field accurate collision detection, even with complicated geometry, is simple, robust and efficient.

These advantages have made level set surface tracking an attractive alternative to explicit methods for free surface fluid animation.

The level set method constitutes a large body of theoretical concepts, equations and numerical methods. The following parts of this section will focus on introducing the basic concepts of level sets in order to provide a foundation for the work presented in this thesis. For a more in-depth study of level sets consider the book by Osher and Fedkiw [2002] as well as the thesis by Nilsson [2009].

Fundamental Theory of Level Sets

The level set method describes the fluid surface $\partial\Omega$ as the level set Γ of a scalar field $\varphi(\mathbf{x}, t)$, i.e.

$$\Gamma = \{\mathbf{x}(\mathbf{t}) \in \Re^d : \varphi(\mathbf{x}, t) = c\} \tag{2.34}$$

In equation (2.34) d is the number of dimensions and c is a typically constant iso-value. Thus the level set Γ can also be considered an iso-surface of φ. Taking the total time derivative of equation (2.34) results in the equations

$$\begin{aligned} &\frac{d}{dt}\varphi(\mathbf{x}, t) = \frac{d}{dt}c \Rightarrow \\ &\frac{\partial\varphi}{\partial t} + \frac{d\mathbf{x}}{dt} \cdot \nabla\varphi = 0 \end{aligned} \tag{2.35}$$

which if we assume that $\frac{d\mathbf{x}}{dt}$ equals the fluid velocity $\mathbf{v}$ result in equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi = 0 \tag{2.36}$$

Equation (2.36) describes the change to the Eulerian parameter φ due to the motion of the Lagrangian points Γ as they are advected by $\mathbf{v}$. This is equivalent to the situation described in chapter 2.1.1 and if desired equation (2.36) can be obtained by means of the material derivative instead.

In addition to equations (2.34) and (2.36) it is also often convenient to enforce the Eikonal equation

$$\nabla \varphi = 1 \tag{2.37}$$

with Γ as boundary condition. The solution to (2.37) will produce a scalar field φ that describes the signed Euclidean distance from every point in space to the closest point on Γ. It is typically also convenient to choose $c = 0$ and divide φ into regions that are inside Γ and regions that are outside Γ based on the definitions

$$\varphi_{inside} = \{\mathbf{x} \in \Re^d : \varphi(\mathbf{x}(t), x) < 0\} \tag{2.38a}$$

$$\varphi_{outside} = \{\mathbf{x} \in \Re^d : \varphi(\mathbf{x}(t), x) > 0\} \tag{2.38b}$$

Thus the sign of φ determines whether a point in space is inside or outside Γ. The closest point $\mathbf{x}_{cp}$ on Γ from some point $\mathbf{x}$ can now be found by means of the closest point transform (CPT)

$$\mathbf{x}_{cp} = \mathbf{x} - \varphi(\mathbf{x}) \nabla \varphi(\mathbf{x}) \tag{2.39}$$

In addition to the convenient collision detection allowed by the distance information encoded in φ signed Euclidean distance fields also allow for easy calculation of several important geometric properties. The surface normal $\mathbf{n}$ of Γ can be calculated through

$$\mathbf{n} = \frac{\nabla \varphi}{|\nabla \varphi|} \tag{2.40}$$

and thus the mean curvature κ can be obtained through

$$\kappa = \frac{1}{2} \nabla \cdot \mathbf{n} \tag{2.41}$$

These properties can be calculated for any level set of φ, i.e. essentially anywhere in space where φ is defined. This property is very convenient when solving volumetric problems such as the Navier-Stokes equations and is in stark contrast to explicit geometry where surface properties are typically only defined on the actual surface.

Disadvantages with Level Sets

Although the level set method has many advantages it also brings with it its own set of problems and shortcomings. This is especially true when φ is sampled on a uniform computational grid as is a requirement for the finite difference schemes commonly used with level sets [Osher and Fedkiw, 2002]. The limited sampling rate of the Eulerian grid puts a limit on the highest spatial frequency that can be represented due to the Nyquist sampling theorem[5]. As a result low resolution level set surfaces can look overly smooth with no sharp corners or thin features beyond the resolution of the computational grid. Furthermore aliasing effects may develop for level set features that are close in size to the distance between two sampling points. Level sets also typically suffer from numerical smoothing of sharp features during advection, slowly eroding away surface details over time. This problem can be reduced by the use of high order accurate numerical advection schemes, for example using the Total Variation Diminishing (TVD) Runge-Kutta [Shu and Osher, 1988] time integration scheme and the Weighted Essentially Non-Oscillatory (WENO) [Liu et al., 1994] spatial integration scheme. The Back and Forth Error Compensation and Correction (BFECC) method presented by Dupont and Liu [2003] can also be helpful for this purpose. However, inaccurate advection remains a significant issue when using level sets for animating and tracking fluid surfaces.

2.4.5 Hybrid Level Set Methods

In addition to purely Lagrangian and Eulerian methods there are a number of hybrid surface tracking methods commonly used for free surface fluid animation. These methods include the Particle Level Set (PLS) of Enright et al. [2002a] and the Marker Level Set (MLS) of Mihalef et al. [2007]. The PLS method adds a large number of small spheres around Γ in order to augment the level set formulation with some of the advantages of Lagrangian particles. The MLS method instead takes a more Lagrangian approach by adding particles to Γ itself. The goal with these hybrid methods is to try and combine the strengths of both Lagrangian and Eulerian methods effectively obtaining the best of both worlds.

The Particle Level Set Method

The Particle Level Set (PLS) method first introduced by Enright [Enright et al., 2002a, 2005] uses level set surface tracking at its core but augments the implicit surface with Lagrangian tracking particles. These particles are passively advected with the level set and used to mitigate some of the issues with level sets as mentioned in section 2.4.4.

[5]For more information on sampling theory consider relevant textbooks such as the book by Mandal and Asif [Mandal and Asif, 2007].

The method relies on maintaining a potentially curvature dependent density of particles in a close neighborhood around the level set surface. Each particle is assigned a position and a signed radius where the radius of the particle represents the distance to the closest point on the implicit surface. The PLS method has proven very effective for use with free surface fluid animation [Enright et al., 2002b, 2003, 2005; Losasso et al., 2008] since Lagrangian particles suffers from much smaller numerical errors during advection than the level set method typically does. However, the particle level set method also has a number of disadvantages. In practice a large number of particles is necessary for the method to be effective and thus the PLS requires a fairly large amount of memory compared to a pure level set [Nielsen et al., 2007; Söderström and Museth, 2010]. Also, as the fluid surface deforms regions may stretch and contract causing the particle density to be too low in some regions and too high in others. In these cases new particles will have to be created and old particles destroyed. The removal of particles also removes the surface information carried with their radius and position information, thus reducing the effectiveness of the method. Likewise creating new particles requires estimating the closest distance to the implicit surface which will typically also introduce errors into the surface representation described by the particles. Finally fluid animation often creates scenarios where the fluid self-intersects. In this case the particle representation may create an inconsistent surface representation and level set based cleanup and re-initialization will be necessary. All these cases can lead to the introduction of surface representation errors that can easily result in surface noise, thus creating a fluid surface that is not visually pleasing. Nevertheless particle level sets can be used to create quality fluid animations as is shown in figure 2.3 as well as figure 3.14 in section 3.4.4.

The Marker Level Set Method

The work presented in this thesis focus primarily on the PLS method (Paper I) or pure level set surface tracking (Paper II and Paper III). However, the particle compression and streaming method presented in Paper I is in theory applicable to any type of hybrid surface tracking approach where particles are stored close to a level set surface. One such method is the Marker Level Set (MLS) method first introduced by Mihalef et al. [2007]. Like the PLS method MLS relies on a level set surface at its core and augments this surface with Lagrangian tracker particles. However, instead of indirectly sampling Γ using particle position and radii as the PLS does, MLS tries to directly sample Γ by placing the tracker particles on the actual fluid surface. As a result the MLS method requires no distance information to be stored by the particles. The implicit surface represented by the particles is then reconstructed using a blending function. This method has the same advantages as the PLS concerning small advection

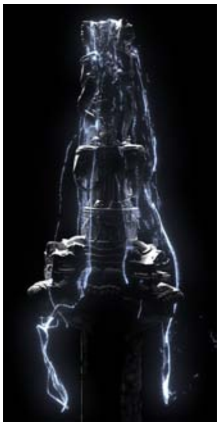


Figure 2.3: A PLS surface tracking example showing a free surface fluid animation of a fountain. This image was created by rendering both the level set surface and the tracker particles.

errors compared to the level set method. However it also suffers from some of the same issues. Surface particle density needs to be maintained and when discrepancies arise between the particle surface representation and the level set one needs to decide to what degree the one or the other should be trusted. The particles are typically more reliable after advection, but this may not be the case in regions where the particle density has been lowered due to surface stretching and where topology changes have occurred. Nevertheless MLS provides an interesting, although currently not widely used, alternative to the PLS method.

2.5 Memory Efficient Eulerian Simulation

Up until this point we have primarily discussed the theory behind Eulerian free surface fluid animation and level set surface tracking. However, in order to turn this theory into practical animations all the relevant equations need to be solved. Since the Navier-Stokes equations have proven notoriously hard to solve analytically for anything but the most trivial of cases, computers and numerical methods are used for computing the actual animations. The core component of an Eulerian fluid animation system is the computational grid onto which all Eulerian quantities are sampled. The simplest method for constructing such a computational grid is to make a uniform discretization of the 3D space in which the simulation takes place. However, since the number of samples need to be finite a limited region of 3D space must be chosen where the fluid animation takes place. This region is typically called the “simulation box” or “simulation domain” since it specifies the boundaries of the 3D world in which the fluid animation can take place.

This local "world" in which the fluid exists can be very limiting for Eulerian free surface fluid animation. Consider as an example an animation of a massive tidal wave flooding the city of New York. Although the water will not flood the whole city at once, and possibly might not even cover a fraction of the city, the computational grid used must be able to cover every part of space where the fluid *might* go. Thus a simple, uniformly sampled computational grid for this task can easily become quite massive. The city of New York has an estimated surface area of $800km^2$ [NYC Department of City Planning, 2010] which can be approximated as a square with the side $28km$. The tidal wave should at least interact with large buildings, thus a sampling rate for the Eulerian parameters of one sample every 5 meters should be sufficient. Assuming that the simulation domain is 50 meters high this results in $5600^2 \times 10 \approx 3 \cdot 10^8$ samples. Thus sampling a single Eulerian parameter, like the fluid velocity, on this fairly coarse domain will require a minimum of 3.8GB[6] of data.

As a result the data requirements for this type of large scale and/or high resolution fluid animations may exceed the amount of fast primary storage, i.e. "computer memory" available. In order to reduce the amount of primary storage required to represent large Eulerian simulation domains a number of different methods has been developed in the fields of computer science and computer graphics. A few of these methods of relevance to this thesis are briefly presented in sections 2.5.1 - 2.5.6 below.

2.5.1 Out-of-Core Computing

When faced with data that is too large to fit in primary storage a common approach is to extend the amount of available memory by also including slower secondary storage devices such as hard disk drives. This is done through an external memory algorithm that typically tries to ensure that data on which computations are currently performed is kept in primary storage, whereas data that is currently not used is transferred to secondary storage. General-purpose algorithms for this approach are included in many popular operating systems and thus many computers will automatically use secondary storage when primary storage is full. This process can however be very inefficient since the operating system generally does not know what data will be used by individual programs or when. As a result significant performance gains can be achieved by creating external memory algorithms that are customized for a specific application. Such algorithms can be aware of the data usage patterns of their application and can thus predict when and how data will be used. Computations that make use of external storage algorithms will be referred to as Out-of-Core (OOC) computing.

Historically OOC computing has been used in a wide variety of areas, examples of which include linear algebra, image processing, relational and

[6]Assuming 3 vector components, each stored as a single precision floating-point value.

spatial databases, simulation and computer graphics among many others. For a recent survey of the field of OOC methods consider the work by Vitter [2001]. A specific survey in the field of linear algebra and simulation is provided by Toledo [1999] and a survey concerning the field of computer graphics is provided by Silva et al. [2002]

2.5.2 Dynamic Rectangular Domains

The example presented in section 2.5 requires a large simulation domain in order to allow the fluid to reach every corner of New York City. However, for free surface fluid animation the Eulerian parameters are only needed where the Navier-Stokes equations are actually solved at the moment. For the New York flooding scenario this corresponds only to the region where there is currently water. As a result memory efficiency can be increased by continuously adapting the size of the original rectangular grid in such a way that it only covers the region that contains water at any given moment. This approach effectively removes any restraints on how the fluid is allowed to move and also reduces the amount of "empty space" stored during free surface fluid animations. However, for complex, concave fluid shapes the efficiency of this type of simple grid is quickly reduced.

2.5.3 The Octree Grid

A more memory efficient approach to the simple dynamic grid presented in section 2.5.2 is the octree grid. By using an octree data structure and collecting the leaf-nodes in and around the actual fluid, a grid structure can be obtained that better matches the actual shape of the fluid region. The octree still needs to store samples in the "empty" space where there is currently no fluid. However, these samples can be stored sparsely, thus reducing the memory requirements of the simulation compared to the simple rectangular grid. Although octrees also allow for non-uniform sampling of the Eulerian parameters this tends to cause problems in practice. The finite difference methods typically used when solving the Eulerian Navier-Stokes and Level Set equations are constructed for uniformly sampled grids and do not respond well to non-uniform sampling. A good example of fluid animation using height-balanced octree grids is the work by Losasso et al. [2004].

2.5.4 The Dynamic Tubular Grid

In order to achieve even higher memory efficiency for Eulerian free surface fluid animation, a Dynamic Tubular Grid (DT-Grid) can be used. The DT-Grid was first introduced by Nielsen and Museth [2006] as an efficient data structure for dynamic level set geometry. However, with minor changes this grid structure can also be used for dynamic volumetric phenomena like water. This Dynamic Volume Grid (see section 2.6.2) was developed as part of the research behind Paper I and is used extensively together with DT-Grids in the work presented in this thesis.

The DT and DV Grids are geometrically adaptive and can change their shape to perfectly match the shape of any level set surface geometry. Due to their projective nature these grids do not need to store any data in empty space and thus come close to offering optimal memory efficiency for a uniformly sampled Eulerian free surface fluid animation system. A related data structure to the DT-Grid is also the Hierarchical Run-Length Encoded (H-RLE) level set method presented by Houston et al. [2006].

2.5.5 The Dynamic Block-Grid

Although the DT-Grid is close to memory optimal [Nielsen and Museth, 2006] the grid relies heavily on data being accessed in sequential order. The performance of the DT-Grid can be significantly decreased when data is accessed in a non-sequential manner. Although all operations necessary to compute free surface fluid animations can be described as sequential iterations through DT and DV grids as was done in Paper I, there are many scenarios where fast random access is desirable. In parallel to the work presented in this thesis Ken Museth has developed an efficient block-based level set data structure, dubbed the DB+Grid[Museth, 2009]. Given the fact that the DB+Grid is still propriety technology, our insight into this data structure is limited. However, for the sake of completeness we include the following high-level description, and refer the interested reader to an upcoming publication.

The DB+Grid is based on a Dynamic Blocked Grid structure that exploits the spatial coherency of uniform grids to effectively separate the encoding of data values and topology. The grid shares several characteristics with the B+Trees typically employed in data-bases and file-systems, which accounts for its name. The DB+Grid allows for cache-coherent and fast data access into sparse 3D grids of very high resolutions, i.e. exceeding millions of grids-points in each spacial direction. Additionally, the DB+Grid is very general since it does not impose topology restrictions on the sparsity of the volumetric data or access patterns when the data are inserted, retrieved or deleted. This is in contrast to both DT and H-RLE grids that assume fixed data topology (narrow-band level sets), and require specific access patterns (sequential) for fast data access. Since the DB+Grid employ a hierarchical data-structure it also offers adaptive grid sampling and native acceleration structures which leads to fast algorithms. As such, DB+Grid has proven useful for several applications that call for very large sparse dynamic volumes, and it has already been featured in the live-action movies "The Golden Compass", "Pirates of the Caribbean: At World's End"[Museth et al., 2007], "The Mummy, Tomb Of The Dragon Emperor"[Museth and Clive, 2008], and "2012"[Zafar et al., 2010].

2.5.6 Adaptive Eulerian Grids

Since the complexity of fluid flow can vary greatly depending on spatial location, an adaptive grid structure for Eulerian fluid animation can be desirable. However, since the finite difference schemes used for fluid animation rely on uniformly sampled grids adaptivity is non-trivial. Adaptivity can still be provided without breaking the uniform sampling constraints of the finite difference schemes by computing finite differences on a uniformly sampled grid while solving the relevant equations on a transformed, adaptive version of this grid. By making use of the Jacobian of the grid transformation the equations described on the adaptive grid can be solved on the uniform, i.e. undistorted grid. This method was used successfully by Tang et al. [2003] for level set deformations and later also by Tan et al. [2007] for fluid simulation. We suspect that these methods can also be successfully applied to free surface fluid animation and this is an ongoing research project at the writing of this thesis.

However, although adaptivity is an attractive property it is not without its own problems. First of all adaptivity through grid transformation rely on a rectangular, uniformly sampled grid and it is not clear how this method can be applied to the arbitrary volumetric grid shapes provided by the DT and DV grids. Thus this method can typically not compete with the memory efficiency of these grids in spite of its adaptive nature. Furthermore, adaptivity brings with it the problem of choosing how the grid should be adapted, i.e. designing a good adaption oracle. In order for adaptive methods to be effective sample points need to be placed in regions where high frequency behavior will arise from the equations being solved. However, since the solution to the equation of interest is unknown until the equation has been solved this problem is essentially a paradox: Perfect adaptivity requires a perfect solution to the equations to be known. However if such a solution already exists adaption is not needed since the problem is already solved. As a result oracles tend to try and guess the solution to the equations being solved and thus predict where sample points should be concentrated. This biased behavior can lead to inaccurate or even completely false solutions to the relevant equations if the oracle prediction is inaccurate and sample points are concentrated in the wrong regions. As a result adaptive computations can, for a general problem with an unknown solution, do more harm than good. Nevertheless adaptive finite difference simulations of Eulerian fluids is an interesting possibility.

2.5.7 MAC Grids

The Marker-And-Cell (MAC) grid by Harlow and Welch [1965] is not really a grid structure per se, however it will be mentioned here since MAC grids are for numerical reasons commonly used for fluid animation. The MAC grid is essentially a regular Eulerian grid specifically designed to store the fluid velocity field. Where regular Eulerian grids store the full velocity vector at every sample point the MAC grid instead stores the velocity components on the corresponding faces between grid cells. Thus the full velocity vector never exists at any one point in space. However the cell-centered velocity vector can be approximated by for example averaging the neighboring velocity components.

The main reason for using MAC grids is a potential problem with the combination of regular grids and the central difference numerical scheme. For the central difference scheme there exists a frequency at which the numerical derivative evaluates to zero while the numerical values of the grid can vary in a checker-board pattern. As a 1D example consider the sequence $\{...2, 1, 2, 1, 2, 1, 2...\}$. A central difference computation at any point along this sequence will evaluate to zero even though the values obviously are spatially different. Central difference computations on a MAC grid only relies only on adjacent values thus avoiding this problem.

More practical information on MAC grids and their use in fluid animation can be found in for example [Bridson, 2008].

2.6 Constructing an Eulerian Fluid Solver

The computational grids presented in section 2.5 allow Eulerian parameters to be discretized in a memory efficient manner. Using this discretized representation the Navier-Stokes and level set equations can be solved through numerical methods. As presented in section 1.1 robustness, reliability and speed are primary concerns when animating fluids for visual effects. One of the popular Eulerian methods for animating incompressible fluids under these constraints is the Stable Fluids method [Stam, 1999]. A modified Stable Fluids solver combined with level set based surface tracking form the foundation for the free surface fluid animation framework used for generating the results presented throughout this thesis and its included papers. In section 2.6.1 below a short outline of the Stable Fluids method is presented. Implementation specific details of relevance for this thesis are then presented in section 2.6.2. For more detailed information on the different steps of the Stable Fluids algorithm, as well as relevant numerical methods, we direct the interested reader to the work by Stam [1999] as well as the fluid simulation book by Bridson [2008].

2.6.1 The Stable Fluids Method

The Stable Fluids [Stam, 1999] method has historically been one of the cornerstones of fluid animation. The popularity of this method mainly stems from the fact that it is unconditionally stable. Thus this method can potentially find solutions to the Navier-Stokes equations regardless of how coarse the spatial and temporal resolution is and regardless of how violent and unphysical the interactions are between the fluid solver and the rest of the scene.

The Stable Fluids method solves the Navier-Stokes equations in parts by using a first order accurate operator splitting scheme. This allows the different terms of the Navier-Stokes equations to be integrated one by one, allowing for specialized solution methods for each integration step. Typically the solution algorithm consists of the following steps: Given the divergence-free velocity field $\mathbf{v}_t$ at some time t, integrate the self-advection equation

$$\frac{\partial \mathbf{v}_t}{\partial t} = \mathbf{v}_t \cdot \nabla \mathbf{v}_t \tag{2.42}$$

Δt seconds forward in time by means of the method of characteristics, i.e. semi-Lagrangian integration. Let $\mathbf{v}_1$ be the result of this integration. Now add the effect of viscosity by integrating the diffusion equation

$$\frac{\partial \mathbf{v}_1}{\partial t} = \frac{\mu}{\rho} \nabla^2 \mathbf{v}_1 \tag{2.43}$$

Δt seconds forward in time by means of first order accurate implicit integration. This amounts to solving a sparse linear equation system that is positive definite. For an arbitrarily shaped fluid domain such systems can be solved in a fairly efficient manner using for example the preconditioned Conjugate Gradient method [Shewchuk, 1994]. Denote the resulting solution $\mathbf{v}_2$. Now integrate the effect of external forces by solving the equation

$$\frac{\partial \mathbf{v}_2}{\partial t} = \frac{\mathbf{f}}{\rho} \tag{2.44}$$

Δt seconds forward in time. Given the fact that $\mathbf{f}$ is independent of $\mathbf{v}_2$ this can be done in an unconditionally stable manner using explicit integration. Let the resulting velocity field be known as $\mathbf{v}_3$. Now project the potentially divergent velocity field $\mathbf{v}_3$ onto its divergence free part $\mathbf{v}_{t+\Delta t}$ by means of the Helmholtz-Hodge decomposition

$$\nabla^2 q = \nabla \cdot \mathbf{v}_3 \tag{2.45a}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_3 - \nabla q \tag{2.45b}$$

thus enforcing equation (2.21b). At this point we have obtained an approximate solution $\mathbf{v}_{t+\Delta t}$ to the incompressible Navier-Stokes equations Δt seconds forward in time. We may now integrate $\mathbf{v}_{t+\Delta t}$ another timestep Δt by starting with equation (2.42) using $\mathbf{v}_t = \mathbf{v}_{t+\Delta t}$. Note that by scaling the scalar field q by $\frac{1}{\rho}$ this field can be interpreted as the internal pressure field p of the fluid. The Stable Fluids method is known to introduce significant amounts of artificial viscosity into the fluid animation due to high levels of

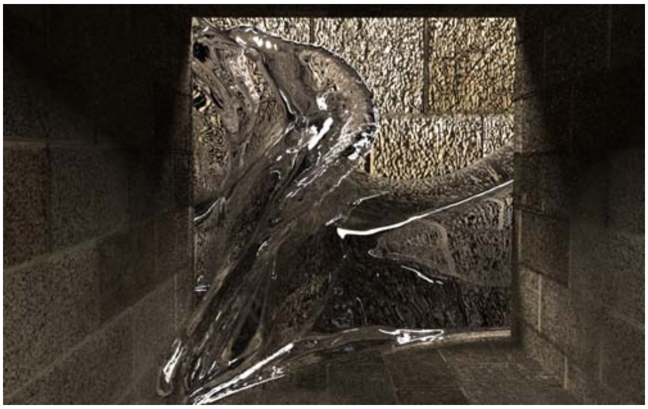


Figure 2.4: An example of the fluid animation quality that can be produced by a fluid solver based on the Stable Fluids approach. This image is a high quality rendering of a frame from the flooding simulation used to test the rendering cluster presented in [Johansson et al., 2006].

numerical dissipation. This is especially true if simple trilinar interpolation is used when solving equation (2.42). However, the robustness and stability provided by the Stable Fluids method still makes it an attractive option and as can be seen in for example figure 2.4 this method can indeed be used to produce visually pleasing animation of incompressible free surface fluids.

Reducing Numerical Dissipation

The high amounts of numerical dissipation introduced by the Stable Fluids method is to a large degree caused by the interpolation necessary as part of the semi-Lagrangian integration of equation (2.42). Numerical dissipation due to interpolation errors can be reduced by means of constrained higher order accurate interpolation like the Constrained Interpolation Profile (CIP) [Yabe et al., 2001] and the Catmull-Rom based interpolation presented by Fedkiw et al. [2001]. A Monotonic Cubic Interpolation [Fritsch and Carlson, 1980] is also an option. However, if large stable timesteps is not a primary concern equation (2.42) can also be solved through high order accurate explicit integration, thus avoiding the use of interpolation altogether. The non-linear aspect of equation (2.42) does however make this a somewhat risky venture. Nevertheless explicit integration of equation (2.42) has been used successfully together with a Courant-Friedrichs-Lewy (CFL) stability condition in both Paper II and Paper III.

2.6.2 Free Surface Fluid Animation on Dynamic Grids

All results presented in this thesis are generated using a free surface fluid animation system based on the Stable Fluids method and level set based surface tracking. To the extent of our knowledge this system is unique in its use of the Dynamic Tubular and Dynamic Volume grids for free surface fluid animation, resulting in several attractive capabilities:

Boundless Animation : The fluid animation system has no inherent “simulation box” limiting the domain in which the fluid can move. The fluid animation is free to go wherever the physics of the situation describes. No decisions need to be taken beforehand on the limits of the “world” in which the fluid exists and its motion is only limited by the physical objects present in the scene. Examples of the behavior made possible can be seen in figure 2.5.

High Memory Efficiency : The compact DT-Grid and DV-Grid (see section 2.6.2) provides a close to optimal memory footprint for a uniformly sampled Eulerian grid for use with free surface fluid animation. As a result our system is able to animate fluids using a minimum amount of memory for representing the fluid domain, the fluid level set and level set based solid boundaries. Our system is also able to use PLS surface tracking, however as is shown in Paper I PLS surface tracking drastically increases the memory requirements of the animation, an issue that was addressed in Paper I and later resolved in Paper II.

Cache Coherency : Our fluid animation system offers good cache coherency, putting small loads on the memory subsystem while attempting to maximize the amount of computations performed on data present in fast cache memory. Overall this is beneficial for performance and also useful for OOC streaming as presented in Paper I.

The Dynamic Volume Grid

Level set operations like advection and re-initialization can be localized in a close neighborhood around the actual implicit surface (i.e. typically the zero level set). These localized *narrow band* computations [Adalsteinsson and Sethian, 1995] allow the computational complexity of level sets to scale with the surface area of the implicit surface instead of the volume of the Eulerian grid. The DT-Grid is a further development of this method that also allows the memory footprint of a level set to scale with the surface area of the implicit surface by only maintaining grid data in the computational narrow band. Although this works well for level sets it is not ideal for free surface fluid animation since a fluid is a volumetric phenomenon. In order to solve the Navier-Stokes equations in the fluid a volumetric equivalent

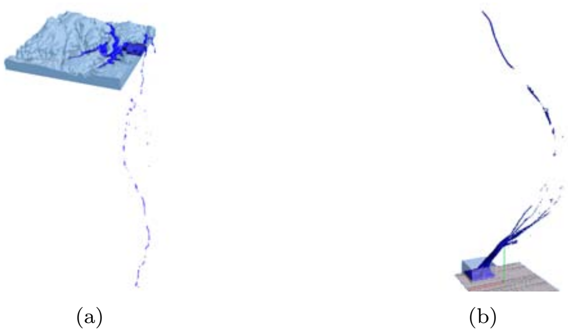


Figure 2.5: An example of the level of freedom allowed by our fluid solver. In figure 2.5(a) a model of a section of the Grand Canyon is flooded. During the animation water escapes the confines of the model and starts free-falling over the edge. In figure 2.5(b) a powerful explosion takes place in a box. The effectively unlimited simulation domain provided by the DV-Grid allows the Eulerian fluid to travel unhindered far outside the originally estimated confines for this animation.

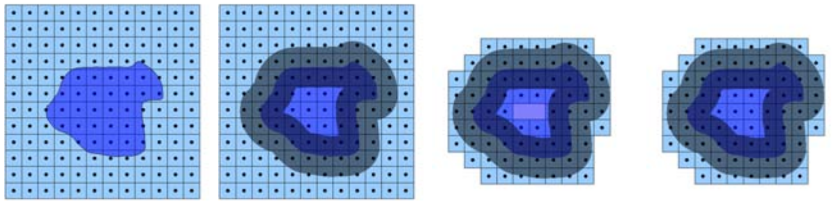
to the DT-Grid is needed. For this purpose the Dynamic Volume Grid (DV-Grid) was constructed. This grid is similar to the DT-Grid in its adaptive properties; however instead of storing only the narrow band the DV-Grid stores the union between the narrow band of the fluid surface and the volume encompassed by the fluid. An example of the difference between a regular grid, the narrow band method, the DT-Grid and the DV-Grid can be seen in figure 2.6.

A DV-Grid can be derived from a DT-Grid in a fairly straight-forward manner requiring only minor modifications and can be considered a close to memory optimal uniform grid for Eulerian free surfce fluid animation.

MAC Grid Computations on DV-Grids

Our fluid animation system supports both MAC Grids (see section 2.5.7) and regular velocity grids where the full velocity vector is stored at the grid center. Velocity data stored on DV-Grids can readily be reinterpreted as velocity component data stored on grid cell faces and no changes to the data structure is needed to enable MAC grid computations.

The results in Paper I, Paper II and Paper III were all computed using regular grids storing the full velocity vector at cell centers. However, to the extent of our knowledge all the methods presented in these papers can readily be applied to fluid animations using MAC grids. Indeed a number of the results presented in this thesis, for example figures 2.8, 2.10, 4.4(b) and 4.10 are fluid animations performed on such staggered grids. The PML boundaries presented in Paper III are also possible to apply to MAC grids,



(a) Regular Grid (b) Narrow Band Grid (c) DT-Grid (d) DV-Grid

Figure 2.6: Examples of different levels of optimization for uniform Eulerian grids. Figure 2.6(a) shows a 2D fluid (dark blue) on a uniform (non-MAC) Eulerian grid. Grid data is stored in the positions marked by the black dots and level set computations are performed everywhere. Figure 2.6(b) shows a narrow band grid where grid data is stored globally whereas computations are localized in a narrow band (darkened region). Figure 2.6(c) shows the DT-Grid where both data and computation is localized. Figure 2.6(d) shows the DV-Grid where the interior volume is added to the DT-Grid.

a topic that is specifically addressed in section 5.8.

Particle Level Sets on DT-Grids

The DT-Grid is designed to access data in sequential order and thus it is natural for all data associated with DT and DV grids to be stored in sequential order as well. Maintaining this sequential storage order is straightforward for Eulerian data that naturally coincides with the location of grid cells. Examples of such data include the level set function φ around which the DT-Grid is adapted as well as fluid velocity and pressure data. This is however not the case when associating Lagrangian data with Eulerian grids as is the case for the PLS and MLS methods. Even if the particles are accessed in sequential order each particle can have an arbitrary position thus resulting in a non-sequential and in the worst case random mapping to the cells in the DT-Grid. Likewise if the DT-Grid is accessed sequentially an arbitrary and potentially random set of particles can be associated with each grid cell. In order to maintain the efficient, cache-coherent storage of the DT-Grid when hybrid surface tracking is used a special particle advection algorithm is used. This algorithm is able to sort the particles in linear time[7] during each particle advection step so that the particle storage order coincides at all times with the DT-Grid storage order. The algorithm makes use of a particle storage data structure that associates a set of particles to each grid cell. This is done by associating a particle "bin" to each cell in the DT-Grid where each bin contains an index into

[7]Assuming that no particle move further than a distance that can be contained by a DT-Grid iteration stencil.

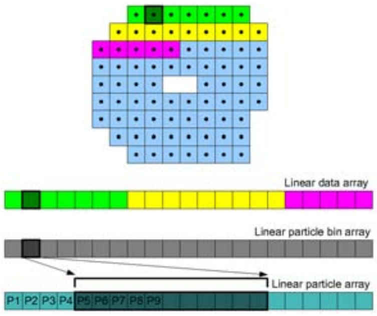


Figure 2.7: An illustration of the DT-Grid to particle array mapping data structure. Each voxel in the DT-Grid has a particle bin associated with it. Each particle bin contains information about the number of particles in the bin and the index to the first particle in the particle array that is within the voxel.

a linear array of particles as well as the number of particles in that cell. An illustration of the data structure is provided in figure 2.7. Using this storage method, particles can be advected while maintaining their DT-Grid aligned storage order through the following algorithm:

Advection : Advect each particle in accordance to the PLS or MLS method by iterating through the DT-Grid in sequential order.

Mapping : Create a new set of empty particle bins. Iterate through the DT-Grid a second time writing the new number of particles in each bin by taking into account particles that have moved into neighboring cells. Once the particle count is known for each cell the corresponding index into the sorted particle array can be constructed by another iteration through the DT-Grid. Note that the new particle bins that have now been constructed map into the sorted particle array which is yet to be created.

Sorting : Create a new particle array of identical size to the current one. Iterate through the DT-Grid copying particles from the old particle array to the new one taking into account the particles that have changed cells. Providing that the particles have moved only a short distance, which is commonly the case in practice, this operation can be done sequentially by making use of a DT-Grid iteration stencil (see [Nielsen and Museth, 2006]) of appropriate size. For this algorithm a 27 point cubical stencil (see figure 3.9) is typically[8] needed.

Note that all parts of this algorithm consist of sequential iterations through the DT-Grid. Also note that this algorithm is presented in its

[8]Assuming advection limited by a CFL condition.

fundamental form and can be optimized by combining the advection and mapping steps. The algorithm presented above is further refined in Paper I in order to allow for fast OOC streaming of particle data - a situation where the cache coherent nature of the DT-Grid and DV-Grid is effective. Using the particle storage method presented above all additional operations necessary for the PLS and MLS algorithms can be performed sequentially as well.

Accurate Self-Advection

As noted in section 2.6.1 our fluid animation system has the option of solving equation (2.42) using explicit integration in order to reduce interpolation related numerical dissipation. Equation (2.42) is an advection type equation where the advected quantity (the velocity) is the same as the quantity that determines the rate of advection (i.e. self-advection). For level set advection the Total Variation Diminishing (TVD) Runge Kutta [Shu and Osher, 1988] time integration scheme together with the Hamilton Jacobi Weighted Essentially Non-Oscillatory [Liu et al., 1994] upwind finite difference scheme has proven to be effective [Enright et al., 2005]. Under the assumption that the advecting velocity field is constant during each self-advection iteration of the Stable Fluids algorithm equation (2.42) can be interpreted as ordinary advection and the TVD Runge-Kutta/HJ-WENO scheme can be considered appropriate. In practice this self-advection scheme has proven effective and we use it extensively as part of our free surface fluid animation system. This approach has also proven useful for integrating other advected quantities as presented in Paper III.

Our self-advection scheme is explicit and thus only conditionally stable. However, we often use high order explicit methods for level set advection (see for example Paper II) and the CFL stability condition associated with the level set has in practice proven adequate also for our explicit self-advection scheme. We have experienced a few cases where our scheme seems to become unstable, however, since these have been isolated incidents further investigation is needed in order to determine whether this was caused by our self-advection approach or other factors. All results presented in this thesis and its included papers were however computed without any stability issues.

2.7 Artificial Conservation of Volume and Energy

The aliasing issues and difficulties with advecting high frequency geometry exhibited by the level set method can lead to geometric features being lost during the animation of a fluid. Due to such numerical errors both mass and energy can be lost during a fluid simulation. If such losses are large enough the discrepancy will become visible to the naked eye and results in a fluid animation that is not visually pleasing. In our fluid animation system we have introduced a simple method for artificially correcting these problems if so desired. The methods presented below are, unless stated otherwise, *not* used for any of the results presented in this thesis or its associated papers since they hide inaccuracies in the fluid animation and may thus produce results that are misleading from a scientific standpoint. However, both methods provide a fast and simple way of creating visually pleasing fluid animations even at fairly low resolution and thus we feel they should be mentioned.

It should be noted that these methods are useful mainly from an aesthetic point of view since they are, although physically motivated, not physically accurate. Mass (or energy) lost in one specific region of the fluid will be reintroduced over the entire fluid, thus causing unphysical transport of energy and mass. This effect may or may not be noticeable to the naked eye depending on the amount of mass or energy lost per unit of time. However, as is shown in this section the methods described below guarantee perfect conservation of mass and total energy at all times and both methods are very easy to implement.

2.7.1 Artificial Volume Conservation

Equations (2.45a) and (2.45b) constitutes a Helmholtz-Hodge decomposition where the fluid velocity field is projected onto its divergence-free part. However, by adding an additional term to the right-hand side of equation (2.45a) the result of the projection will not be divergence-free. This can be exploited in order to compensate for lost volume. By evenly distributing sources in the remaining fluid volume lost mass can be artificially injected in every grid cell. Since parts of the fluid with large volume contain more grid cells the bulk of the new mass will be added there, thus causing a minimal visual distortion as long as the amount of lost mass is small. Using the same method for subtracting excess mass is however not advisable since the sinks created can easily remove parts of fluid that represents small volumes like thin features and sheets. Potentially this problem can be avoided by making use of the feature thickness estimate provided by the method presented in Paper II. However, avoiding volume subtraction also removes the risk of potentially unstable volume oscillations.

The modified version of equation (2.45a) becomes

$$\nabla^2 q = \nabla \cdot \mathbf{v} + \epsilon_v \tag{2.46}$$

where ϵ_v can be calculated through

$$\epsilon_v = \alpha \left(V_{target} - V_{current} \right) \tag{2.47}$$

Here V_{target} is the desired, for example initial, fluid volume and $V_{current}$ is the current fluid volume. The parameter α is used to determine the rate at

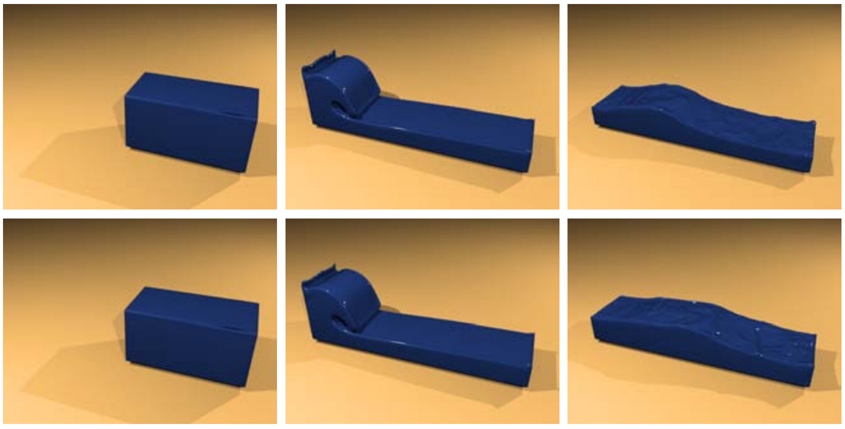


Figure 2.8: Frames from a breaking dam animation where mass loss is present. The top three frames shows the reference animation and the bottom three frames shows the same animation with the artificial volume conservation enabled.

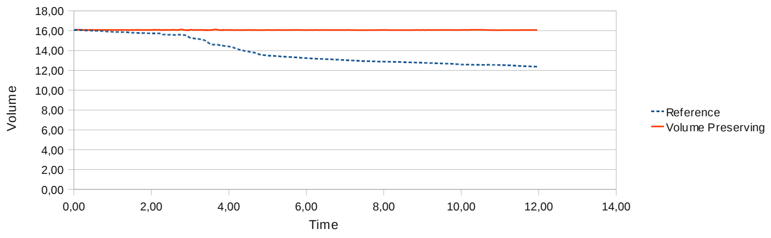


Figure 2.9: Graph showing the fluid volume of the two breaking dam simulations presented in figure 2.8.

which lost volume is recovered. Typically α is continuously estimated such that all lost volume is restored within one or a few iterations of the fluid solver depending on the amount of volume loss[9]. Equations (2.46) and (2.47) can be considered a simple proportional regulator that compensates for lost volume. Since the amount of volume lost is typically small for each iteration of the Stable Fluids algorithm the above method has proven surprisingly effective in maintaining fluid volume without introducing visually noticeable artifacts. An example of the effectiveness of this method is shown in figures 2.8 and 2.9.

[9]Restoring large amounts of lost volume in a short time is possible but can lead to visual artifacts.

2.7.2 Artificial Energy Conservation

If volume and thus mass conservation can be guaranteed, for example through the method presented above in section 2.7.1, conservation of total energy can also be artificially enforced in a simple, although approximate, manner: The total energy E of an incompressible fluid at constant temperature and of constant mass can be described as

$$E = E_k + E_p \tag{2.48}$$

where E_k is kinetic energy and E_p is potential energy. If energy is conserved E will be constant and equal to the initial energy $E_{initial}$ of the fluid. Thus the energy loss (or gain) ΔE in the fluid can be described as

$$\Delta E = E_{initial} - (E_k + E_p) \tag{2.49}$$

Since mass is conserved any energy difference detected through equation (2.49) can be assumed to be caused by inaccurate transformation of potential energy to kinetic energy or vice versa. Energetic flows are typically considered visually pleasing and thus we will restore all lost energy through the addition of kinetic energy to the flow.

The total current energy E can be changed to match $E_{initial}$ by multiplying E_k by the factor α_e where

$$\alpha_e E_k + E_p = E_{initial} \Rightarrow \tag{2.50a}$$

$$\alpha_e = \frac{E_{initial} - E_p}{E_k} \tag{2.50b}$$

Since the kinetic energy E_k of the fluid region Ω can be calculated through

$$E_k = \frac{1}{2}\int_\Omega \mathbf{v} \cdot \mathbf{v} d\Omega \tag{2.51}$$

the velocity field $\mathbf{v}_{new}$ with the associated total energy $E_{initial}$ can be obtained from the current velocity field $\mathbf{v}$ with the associated total energy E through the simple scaling

$$\mathbf{v}_{new} = \sqrt{\alpha_e}\mathbf{v} \tag{2.52}$$

The energy conservation scheme above will produce visually pleasing results if the amount of energy lost per unit of time is small. If a large amount of energy is quickly lost the result will be apparent unphysical acceleration of fast moving parts of the fluid since energy lost in one part of the fluid will be distributed over the entire fluid. Nevertheless, the above method can guarantee energetic flows (see figures 2.10 and 2.11), a property that may or may not be perceived as visually pleasing depending on the scenario. Note that this method does not suffer from the pseudo-unstable behavior of cascading vorticity that can occur when energy is reintroduced

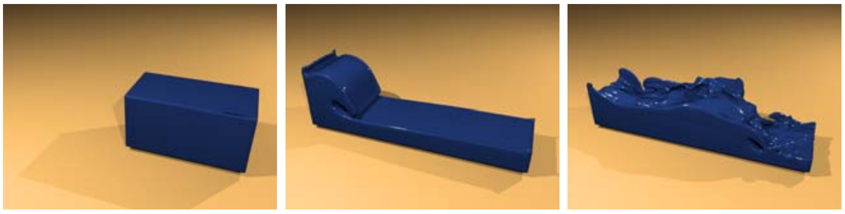
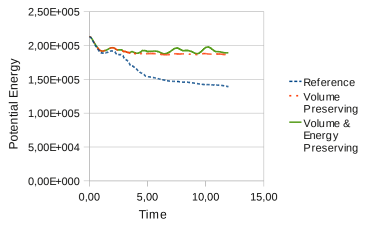


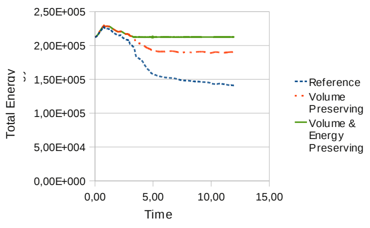
Figure 2.10: Frames from a breaking dam animation where both the volume conservation scheme of section 2.7.1 and the energy conservation scheme of section 2.7.2 has been applied. The frames represent the state of the animation at 0, 2.3 and 8.9 seconds respectively and are comparable to the frames shown in figure 2.8.

with the vorticity confinement method. However, our energy conservation can be combined with vorticity confinement, thus allowing control also of the distribution of rotational energy in the flow.

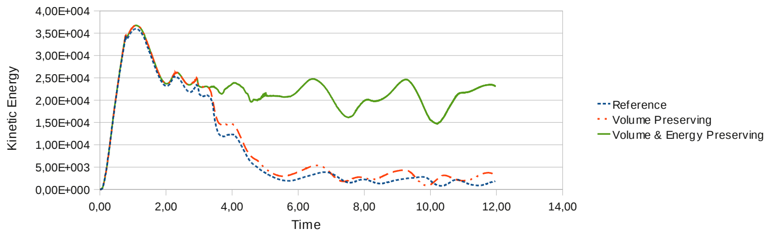
Note that the above method will, in its present form, prevent the fluid from ever coming to rest. It is thus most appropriate for animating completely inviscid fluids, i.e. when solving the Euler equations. However the method can potentially also be used for animating viscous fluids by compensating for the energy changes introduced through equation (2.43).



(a)



(b)



(c)

Figure 2.11: Graphs comparing the potential, kinetic and total energies of the breaking dam simulation depicted in figures 2.8 and 2.10. The reference simulation uses no artificial conservation schemes, the volume conserving simulation uses the scheme presented in section 2.7.1 and the volume and energy conserving simulation uses the schemes presented in sections 2.7.1 and 2.7.2. The initial increase of the total energy is due to over-estimation of the energy of the thin fluid sheet that initially develops along the floor of the box used to contain the fluid.

Chapter 3

Breaking the Memory Barrier - Out-of-Core and Compressed Level Set Methods

This chapter focuses on the novel framework for out-of-core streaming and compression of Eulerian level sets and free surface fluid animations as presented in Paper I. The main focus is the fluid animation aspects of this paper, however, much of the chapter is also relevant for general level set animations. The chapter starts with section 3.1 providing an introduction to the problems that lead to the development the framework presented in Paper I. Next a brief background to this work is provided in section 3.2. This is followed by sections 3.3 and 3.4 describing the central compression and data management components of the framework. The chapter is concluded in section 3.5 with a summary of our framework.

3.1 Introduction

As presented in section 2.4.4 level set geometry has many attractive properties that make it highly useful for free surface fluid animation. However, the finite difference schemes typically used for solving the level set equations favors a uniformly sampled computational grid. Using grid transforms (see section 2.5.6) non-uniform sampling for finite difference schemes can be achieved, however this is currently not widely used in the field of computer graphics. As a result the typical level set implementation for use in fluid animation is based on uniform sampling. The uniform grid structure makes it inherently difficult for level sets to represent high frequency geometric features like sharp corners, edges and thin sheets, all of which commonly occur during free surface fluid animation. Hybrid Eulerian methods, like the PLS, can reduce these issues. However, they are still unable to avoid the fundamental problem that the sampling resolution of the level set is often not enough to represent the desired level of detail. Simply increasing the resolution of the level set surface is also problematic since high resolution level sets and particle level set require a relatively large amount of data. This is especially true if similar level of detail is desired as can be provided by explicit, naturally adaptive alternatives like meshes. Even with state-of-the-art dynamic grids like the DT-Grid the memory requirements of level sets and especially particle level sets can be high compared to meshes.

In order to reduce the data requirements of high resolution level set and PLS geometry Paper I proposes the use of specialized on-line[1] compression schemes for DT-Grid level sets. The goal with these schemes is to be able to significantly reduce the data size of DT-Grid level sets while still maintaining high performance for any numerical computations performed on the level set surface. In order to further reduce the memory footprint of level set and PLS geometry a specialized external memory algorithm for out-of-core computing on DT-Grids is also proposed. During a free surface fluid simulation this method can effectively move level set data between fast but limited primary storage, i.e. "computer memory", and slower but significantly larger secondary storage devices like hard disk drives. As a result of this Out-of-Core (OOC) streaming, essentially all memory limitations on the size and resolution of level set and PLS geometry is removed. By also applying our external memory algorithm to the entire free surface fluid animation system, fluid animations of virtually any resolution can be produced using small amounts of primary storage.

3.2 Background

The use of out-of-core and compression techniques for free surface fluid animation was inspired by the success of streaming and compression methods within the field of computer science. However, no existing methods were found that was well suited for free surface fluid animation on DT-Grids. Thus the decision was made to develop an out-of-core and compression framework specialized for fast and memory efficient level set computations in general and free surface fluid animation in particular. A thorough review of existing out-of-core and compression methods related to our work is presented in section 3 of Paper I and we direct the interested reader to this paper.

3.3 Compression for DT-Grid Based Free Surface Fluid Animation

At its core (no pun intended) our OOC and compression framework consists of one data management component and one compression component. Figure 3.1 shows an illustration of this framework relevant for DT-Grid level sets. In this section the ideas behind the different compression schemes for DT-Grid level set and PLS data will be presented. Each method will be described briefly, however, relevant in-depth descriptions can be found in sections 6 and 7.1 of Paper I.

[1]Compression and decompression is performed transparently on the fly on level set data as it is being processed.

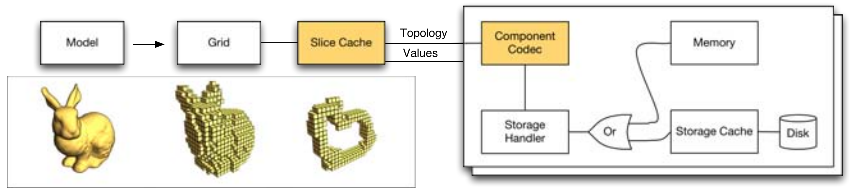


Figure 3.1: An illustration of the out-of-core and compression framework for DT-Grid level sets. The colored boxes represent the compression components while the Storage Handler is responsible for the out-of-core streaming.

3.3.1 Compression of DT-Grid Level Sets

The DT-Grid data structure effectively removes the need to store data and grid topology outside a narrow band (or tube) around the fluid surface (see figure 2.6 in section 2.6.2). This results in a compact Eulerian grid representation that is close to optimal for a uniformly sampled computational grid. Since the DT-Grid is adapted around a level set surface the grid typically contains at least the sampled level set distance function φ in addition to topology data. Thus compression of a DT-Grid typically involves compressing both the level set data and the grid topology. The DT-Grid topology consists of coordinate data of the first and last voxel[2] in a *connected component*[3] with pointers in between. As a result the data layout of the DT-grid forms geometric shapes corresponding to different levels of projections of the geometry of the model. An illustration of this behavior is shown in figure 3.2. The geometric nature of the topology data lends itself well to compression, however, the compression scheme needs to be specialized for each level of projection. If we start by looking at the pointer data of the 1D projection level, an analysis shows that the data tends to be fairly uniform. Thus the difference between consecutive values are almost constant (see figure 3.3). To compress this data a second order accurate probability model [Salomon, 2007] is used.

This characteristic behavior is also present at the 2D projection level as can be seen in figure 3.4, and thus the same probability model used at the 1D level can also be used for compressing pointer data on the 2D level. In addition to the 1D and 2D pointer data we also need to compress the coordinate data stored in the green voxels (see for example figure 3.2) at each projection level. This data is located at the beginning and end of each connected component of the DT-Grid. The coordinates stored are the coordinates of the grid boundaries at each projection level, i.e. if the 1D projection stores the x coordinate, the 2D and 3D projection levels store the y and z coordinates respectively. The grid locations of coordinate data will

[2]In the context of this chapter a voxel is, unless stated otherwise, a cubic grid-cell with the grid data stored at the center of the cell.

[3]See [Nielsen and Museth, 2006] for details on the DT-Grid data structure.

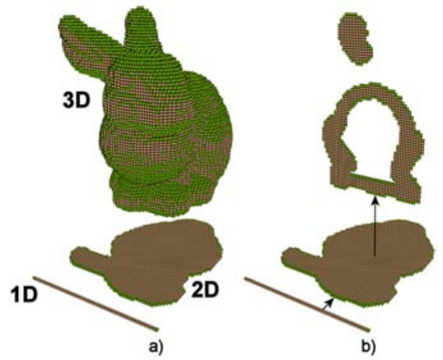


Figure 3.2: a) Illustration of the 1D, 2D and 3D components of a DT-Grid representation of the Stanford bunny at 64^3 resolution. b) The 1D and 2D components contain pointers to columns in respectively 2D and 3D, and the 3D DT-Grid component stores the actual distance values of the level set function. Voxels that contain coordinate data are colored green and voxels containing pointer data is colored brown.

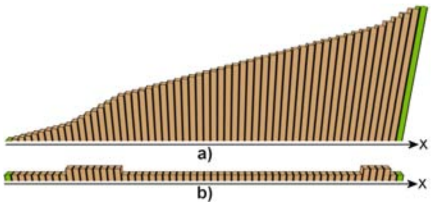


Figure 3.3: a) The values of the 1D projection level shown as a histogram. b) The difference between consecutive values at the 1D projection level shown as a histogram.

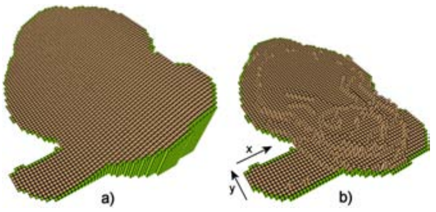


Figure 3.4: a) The values of the 2D projection level constituent shown as a 2D histogram. b) The difference between consecutive values on the 2D projection level shown as a histogram.

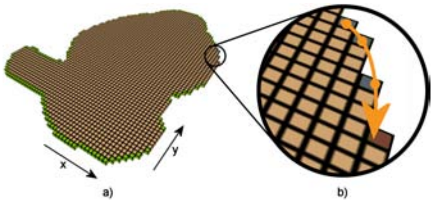


Figure 3.5: a) The 2D grid component of the 64^3 Stanford bunny DT-Grid. The y-coordinates stored at the 2D projection level are shown in green. b) A close-up shows actual y-coordinate (red-brown) predicted by three previous y-coordinates (blue).

thus form surfaces in 3D, contours in 2D and points in 1D, all of which reflect the shape of the geometry stored in the DT-Grid. At the 1D level encoding this data is straight-forward. The difference between coordinate values are encoded using a zeroth order adaptive probability model [Salomon, 2007]. At the 2D projection level coordinate values are estimated using, at the most, the three previously processed coordinate values. Here we use as predictor the Lagrange form of the unique interpolating polynomial [Kincaid and Cheney, 1991] that passes through the previously processed coordinates. An illustration of this process is shown in figure 3.5.

Compressing the z coordinate at the 3D projection level is the most complex operation. Here the predictor used is that each of four immediately neighboring coordinates (see figure 3.6) lie on the same plane. Thus $z(D)$ is predicted as $z(A) + \nabla z\mid_A \cdot \binom{1}{1} = z(B) + z(C) - 2z(A)$, using a one-sided first order accurate finite difference approximation for computing the gradient. Given the permutation-symmetry of this expression with respect to $z(B)$ and $z(C)$ we then compress the prediction using a context-based adaptive probability model (see e.g. [Taubin, 2002]) with the integer value $z(B) + z(C) - 2z(A)$ as context.

At this point the only remaining component to compress is the level set distance field. These values correspond to a large majority[4] of the data stored in a DT-Grid. Thus a fast method for achieving high levels of compression is desired for these values. Here a combination of different techniques is used depending on the location of the predicted value in the narrow-band structure of the DT-Grid (see figure 3.7):

1. If all the blue grid points exist in the narrow band, the value at the red grid point is predicted using the 3D Lorenzo predictor of Ibarria et al. [2003].

2. If some of the blue grid points do not exist in the narrow band the parallelogram predictor of Touma and Gotsman [1998] is used if possible. This can be done if the red grid point is part of a face (four

[4]Typically 80% or more as can be seen in Table V in Paper I.

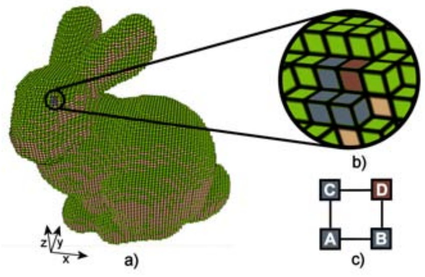


Figure 3.6: a) The 3D grid component of the 64^3 Stanford bunny DT-Grid. The z-coordinates stored at the 3D projection level are shown in green. b) A close-up shows actual z-coordinate (red-brown) predicted by three immediately adjacent z-coordinates (blue). c) The situation in b) shown from above.

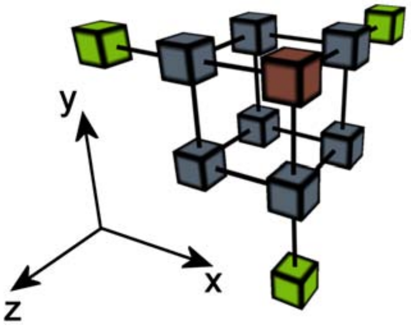


Figure 3.7: Illustration of the grid neighberhood around a level set value (red box) that will be predicted.

connected grid points in the same plane) where all grid points have already been processed.

3. If it is not possible to find a face as described above, an axis-aligned second order differential prediction is used (see Paper I). This essentially constitutes a measure of the acceleration of the values of the level set function along one of the coordinate (i.e. Cartesian) directions.

4. If it is not possible to apply the second order differential prediction a first order differential prediction is used if the previous grid point in one of the coordinate directions exist.

5. Finally, if none of the above conditions apply the value itself is stored.

In order to further increase the compression ratio of the level set distance field we optionally employ quantization of the distance data. In Paper I 14 bit quantization is typically used.

Using the above compression method the memory footprint of DT-Grid level sets can be reduced by between 75% and 92% depending on the specific model and resolution (see tables IV and V in Paper I). Although our level set compression framework achieves high levels of data compression it is fairly computationally expensive. As presented in Table II in Paper I the computational time required for a sequential iteration through the DT-Grid increases by up to a factor of 14. Since performance is a significant concern for fluid animation it may be advantageous to only rely on OOC streaming (see section 3.4) of geometry during the simulation and use the compression scheme primarily for compact long-term storage and generally static level set geometry such as solid boundaries. As is explained by section 8.1.3 in Paper I our compression method is effective also for this scenario. Note however that our compressed level sets, in spite of the performance penalty, allows for significantly higher resolution[5] than otherwise possible given a limited amount of memory. Equivalently this also releases up to 92% of the memroy used by the level set for other purposes, such as solving the Navier-Stokes equations.

3.3.2 Compression of Particle Data

When using the PLS method a large amount of data is needed to store the PLS tracker particles in addition to the level set geometry. In fact, when DT-Grids are used particle data typically becomes the largest data field in the fluid solver by a fair margin (see figure 3.8).

As a result large amounts of primary storage can be saved by compressing the particles. However, in order to maintain performance of the fluid solver such compression also needs to be fast. Our solution to this problem is

[5]Between 4 and 12 times higher resolution based on 75-92 percent compression.

based on the observation that tracker particles are always found within a close neighborhood of the fluid surface and is thus within the grid structure stored by the DT-Grid. Consequently we can make use of the topology data already stored in a highly efficient manner by the (possibly compressed) DT-Grid.

Compression is achieved by describing particle positions and radii in a coordinate frame that is local to the grid-cell in which the particle is currently found. Thus global coordinate information is carried by the DT-Grid and local coordinate information is carried by the particle. This in turn allows us to quantize the particle position and radius in the local coordinate frame - achieving high levels of (lossy) compression without significant performance penalties. Since the quantization is based on the local coordinate frame the error introduced by the quantization will still scale linearly with the size of the grid cell, i.e. if the resolution of the DT-Grid is increased by a factor of two the size of the grid cell halved and the quantization error is halved as well.

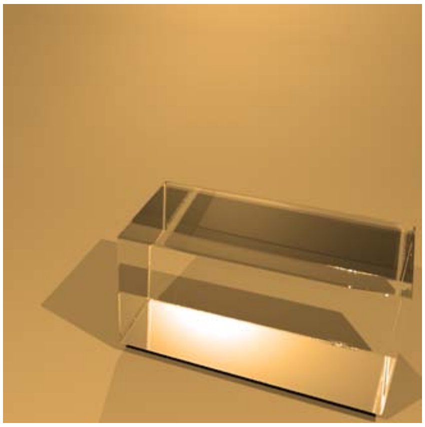
Through experiments we found that a quantization precision of 10 bits for the particle coordinates and 9 bits for the particle radius provided visually pleasing results while still reducing the size of each PLS particle by approximately 70%[6]. Though our tests were performed for PLS surface tracking we believe our quantization method can also be used for MLS surface tracking. However, since MLS stores particles on the actual surface the quantization accuracy of the particle positions may have to be increased. Assuming that 10 bit quantization is enough our compression method would however provide close to 70% compression for MLS particles as well.

In section 9.2 of Paper I we have benchmarked level set advection using our compressed PLS method. For this test the DT-Grid itself was not compressed. As a result we found that our particle quantization method provides a 65% reduction of the simulation memory footprint at a 5% increase in simulation time (see Table VI and VII in Paper I). For the cases shown in figure 3.8 the corresponding reduction of the memory footprint is 63% for figure 3.8(a) and 65% for figure 3.8(b).

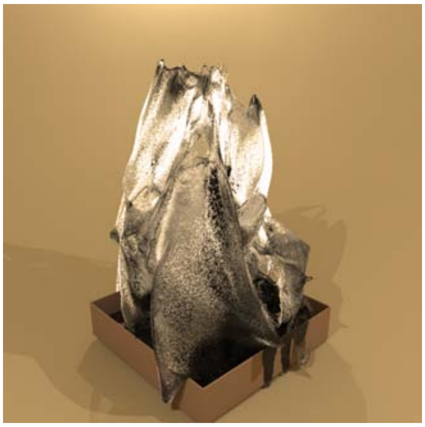
3.3.3 Compression of Fluid Solver Data

The primary focus of Paper I is to facilitate memory efficient and possibly high resolution level set geometry and free surface fluid animations by compressing level set and PLS data, thus increasing the amount of memory available when solving the Navier-Stokes equations. A natural extension of our framework is to also apply on-line compression to the various data fields of the Stable Fluids method. A brief investigation into compression of general scalar and vector fields was conducted as part of this project,

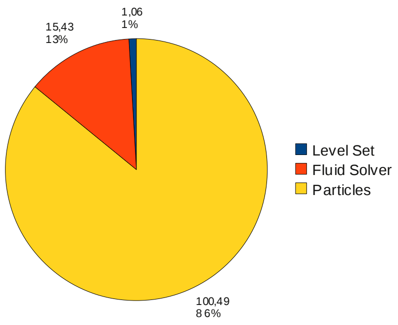
[6]Assuming that particle radius and position was originally stored using single precision floating-point data and that the particle mapping data structure (see section 2.6.2 and Paper I) is uncompressed.



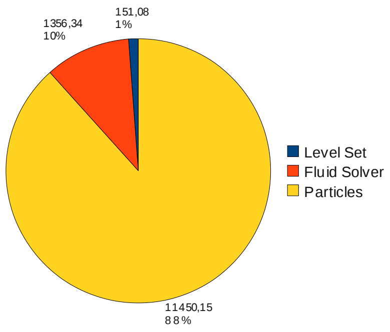
(a)



(b)



(c)



(d)

Figure 3.8: Graphs showing the amount of data required by the different parts of the fluid solver presented in section 2.6. Figure 3.8(c) shows the current data distribution for the simulation depicted in figure 3.8(a). Figure 3.8(d) shows the current data distribution for the simulation depicted in figure 3.8(b). The simulation in figure 3.8(a) has low surface area compared to fluid volume wheras the simulation depicted in figure 3.8(b) has high surface area compared to fluid volume. The memory footprint for each part of the fluid solver is given in MB and as a percentage of the total memory footprint.

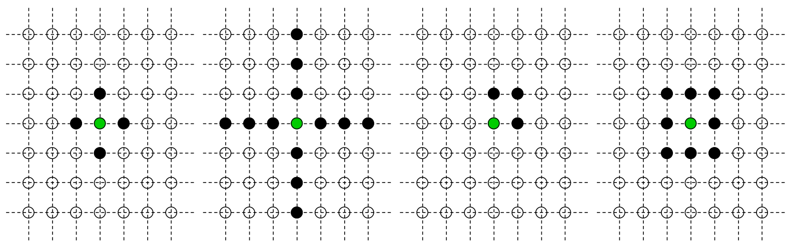

(a) Cross stencil (b) WENO stencil (c) Voxel stencil (d) Box stencil

Figure 3.9: 2D Illustrations of the primary (spatial) data stencils used for fluid animation on DV-Grids. The central grid point of the stencil is highlighted in green. The "cross stencil" (figure 3.9(a) is used for central and first order upwind finite differences. The "WENO stencil" (figure 3.9(b) is used for high order finite differences. The "voxel stencil" (figure 3.9(c) is used for trilinear interpolation. The "box stencil" (figure 3.9(d) is used for higher order interpolation and particle advection (see see section 7.2 in Paper I).

however no satisfactory method for fast general-purpose compression of floating-point data was found at the time. Thus, when necessary our original framework instead relies on OOC streaming (see section 3.4) of the scalar and vector fields when solving the Navier-Stokes equations.

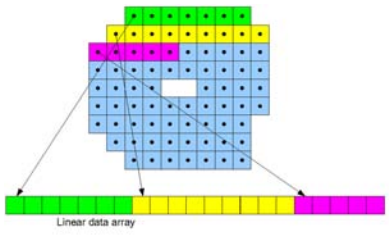
However, since the publication of Paper I a fast compression method for sequential floating-point data has been proposed by Burtscher and Ratanaworabhan [2007]. This method promises high performance and good compression ratios and is thus a promising candidate for compressing velocity and scalar fields in our framework. We do however still expect performance penalties to be high compared to the uncompressed case and thus the practical usefulness of such compression may be limited for simulation.

3.4 Efficient Out-of-Core Streaming for DT-Grid Based Fluids

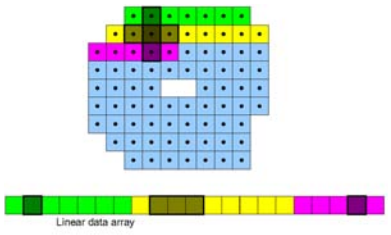
Compression allows us to animate free surface fluids and particle level sets using less memory than previously possible using DT-Grids. In order to allow for even more efficient use of primary storage we also employ a specialized scheme for out-of-core streaming of data on DT and DV grids.

When a computation is performed on data stored on a DT and DV grid a stencil[7] is chosen and then iteratively moved through the entire grid in the lexicographical storage order of the DT-Grid. This ensures that

[7]See figure 3.9 for examples of common stencils used in our fluid animation system.



(a)



(b)

Figure 3.10: Figure 3.10(a) shows an example of how 2D grid data is mapped to a linear array. Figure 3.10(b) shows how this mapping translates to different access positions in the linear array for different points in an access stencil. As the stencil moves sequentially through the grid each acess position sweeps sequentially through the array.

all points in the stencil access data in sequential order, thus providing good data coherency. Since stencil sweep operations only work with a small amount of local data at any time this access pattern lends itself well to out-of-core computing where only the data currently accessed by the stencil needs to be kept in primary storage. Data outside the stencil can then be sequentially read and written to the significantly larger, although typically much slower, secondary storage devices[8] present in most computers. Since secondary storage capacity is typically orders of magnitude larger than primary storage[9] this approach allows computations on implicit geometry of virtually limitless[10] resolution. This method also means that most of the data associated with surface tracking and scene geometry can be moved out-of-core leaving a substantial amount of resources for use with the rest of the fluid solver. However, even though each stencil point access data in sequential order, the access locations will be partially spread over the sequential data array maintained by the DT-Grid. As can be seen from the 2D example in figure 3.10 this leads to a non-trivial although piecewise sequential data access pattern. Consequently an optimal scheme for out-of-core streaming of DT-Grid data is non-trivial to construct. However, since the access pattern is known beforehand for each stencil this is indeed theoretically possible. This knowledge allows us to construct a page replacement scheme for DT and DV grids that is superior[11] to typical general-purpose page replacement algorithms.

[8]This can for example be a hard disk drive.

[9]At the writing of this thesis typical primary memory modules can store 4GB of data while a hard disk drive can store 2-3TB.

[10]The possible resolutions are so high that for all practical purposes the simulation is completely limited by computation time and not by memory.

[11]See the results in section 8 of Paper I as well as the discussion in section 5 Paper I.

3.4.1 Measuring Out-of-Core Computing Performance

In Paper I several performance benchmarks are presented for out-of-core computing on level sets. The result of such benchmarks will depend on the hardware used for computing said benchmarks. Mainly the performance difference between the primary and secondary storage devices are of importance. For the specific hardware used for each benchmark see relevant sections in Paper I. However, note that standard consumer grade devices were used for all tests. The intent is to provide a lower bound[12] on the performance of our out-of-core scheme.

3.4.2 A Page Replacement and Prefetching Strategy for Data on DT-Grids

In theory the optimal page replacement strategy [Tanenbaum, 1992] for a demand-paged system (i.e. no prefetching) is simple: If a page must be evicted from the cache, the page replacement algorithm always picks the page that will be used furthest into the future. Realizing this strategy in practice does however require complete knowledge of the page demand sequence, information that is effectively impossible to obtain in general. However, for specific problems like sequential iteration through a DT-Grid this information is known, allowing the construction of a near-optimal page replacement strategy. Since all computations in our fluid solver can be described as sequential iteration through a DT-Grid or DV-Grid this allows for near optimal[13] out-of-core free surface fluid animation using our solver.

General-purpose page replacement schemes like the Least Recently Used (LRU) scheme can easily make bad decisions when faced with the stencil access patterns used for fluid animation on DT and DV grids. An example of how this may happen is depicted in figure 3.11. This figure shows a 2D grid, a stencil consisting of five points (i.e. the “cross stencil”) and the corresponding positions on the paged secondary storage device. At this point it is important to note that a DT-Grid stencil does not move as one unit. In order to move a stencil one step forward in the lexicographical storage order of the DT-Grid each stencil point is individually moved one step until all stencil points arrive at the new position. Let us now assume that page five is the least recently used page. When stencil point four moves forward it will generate a page fault as page eight is not in memory. As a result page five, being the least recently used, is evicted to make room for page eight. However, as the entire stencil moves, stencil point one will soon move forward into page five which has just been evicted. This generates a new page fault and page five has to be loaded again. Similar to the

[12]More advanced hardware such as professional high-speed hard drives or Solid State Drives (SSDs) possibly in a RAID configuration will drastically increase the data throughput to secondary storage, thus increasing performance.

[13]See table I in Paper I.

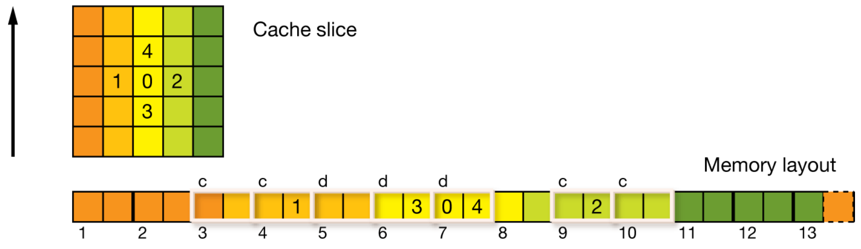


Figure 3.11: Outline of a stencil consisting of five grid points (0-4) as well as a cache slice and the corresponding paged memory/disk layout. In this example all grid points are occupied with data and each page is two grid points long. The pages in memory are outlined in white and the others in black. *c* denotes a clean page i.e. a page that has not yet been written to and *d* denotes a dirty page, i.e. a page that has been written to. Reprinted from Paper I.

LRU strategy it is possible to construct examples where all other existing non-analysis based page-replacement strategies that we are aware of will fail. Analysis based algorithms on the other hand face other problems such as the fact that they need to detect a certain access pattern before they start working properly whereas our approach is guaranteed to work almost optimally at all times.

Our strategy is based on the observation that, although stencil points that are neighbors in space may access data that is far apart all points in the stencil move forward at the same speed. Given this observation the optimal page replacement strategy is to see if the page in memory with the lowest page-id (see figure 3.11) is not part of the stencil. If this is the case the page can be evicted from the cache and, if dirty, written to secondary storage.

In order to keep the secondary storage device busy with I/O operations at all possible times our page eviction strategy is complemented with a page prefetching algorithm. Prefetching is performed by a high priority I/O thread within the Storage Cache. The use of a separate I/O thread allows I/O operations to be performed in parallel with the fluid computations, hence increasing efficiency. Furthermore the I/O thread can exploit direct memory access (DMA) in order to reduce the amount of computations and primary storage bandwidth required per I/O operation, thus leaving as much as possible of these resources to fluid computations.

Our prefetching algorithm works by first checking if all pages that will be accessed by the stencil points are already in-core, i.e. typically all pages immediately ahead of a stencil point. If this is the case no prefetching needs to be done. Prefetching a page typically results in the eviction of a page in order to make room for the new data. This is done in accordance to our previously presented page eviction strategy. To determine which page should be prefetched we use a variation of the elevator algorithm [Tanenbaum, 1992]. Our elevator algorithm always moves in the forward direction, starting at the position of the stencil point corresponding to the lowest page ID. In figure 3.11 this would be stencil point one at page index four. The algorithm then fetches the nearest page in the forward direction from this point that is not currently in-core. The algorithm then moves to the next stencil point, stencil point three in figure 3.11 and so on until it reaches stencil point two. The algorithm then wraps around and starts at stencil point 1 again.

If no page needs to be prefetched the algorithm instead tries to write to disk any dirty pages that will not be written to again during the stencil sweep through the DT-Grid. If no read or write operations need to be performed the I/O thread sleeps until any part of the stencil moves into a new page.

Thus, at any moment in time the I/O thread performs one of the following three steps in order of priority:

1. Prefetching
2. Write-Back
3. Idle mode

In order to test the effectiveness of our paging algorithm a test case consisting of a single WENO stencil pass over an implicit model of the "Stanford Bunny" (see figure 3.12) was performed. The effective resolution of the model is 1000^3 grid-points. In table 3.1 we compare our method to the LRU approach, both methods with and without prefetching. We also compare the results to an optimal page replacement strategy [Tanenbaum, 1992] for a demand-paged system (i.e. no prefetching) as computed from a logged sequence of demand requests. As can be seen our approach comes close to the optimal page-hit ratio for demand paged systems when prefetching is not enabled and we also come close to the optimal page hit ratio of one for prefetched systems. The test in table 3.1 uses relatively small page sizes. In order to achieve high data throughput larger page sizes are typically needed. Figure 3.13 shows the number of grid points processed per second depending on page size and number of pages. As can be seen a page size of 4-8 MB and 32-64 pages gives good results for this test. For the benchmarks presented in Paper I 32 pages of size 4MB were used, showing that efficient out-of-core computations can be performed with our method using as little as 128 MB of primary storage.

3.4.3 Out-of-Core Level Sets for Fluid Animation

Using the prefetching and page-replacement scheme described in section 3.4.2 the level set equation (2.36) can be solved completely out-of-core. Likewise out-of-core re-initialization can also be performed by solving

Figure 3.12: An implicit (level set) model of the "Stanford Bunny".

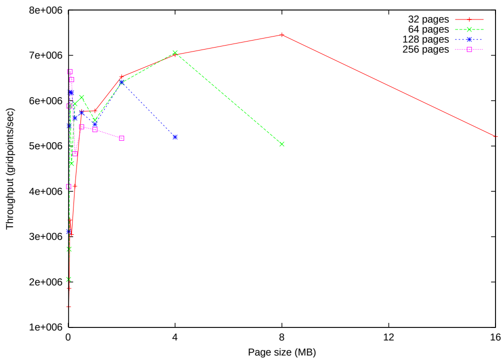


Figure 3.13: The throughput (processed grid-points/second) as a function of page size (in MB) for various numbers of pages in the cache. The results were generated by sequential iteration over the Stanford Bunny in resolution 8000^3 with the "cross" stencil. The maximal memory usage of the cache was in this case restricted to 512 MB.

equation (2.37). This can effectively be used for fluid animation in order to represent huge scenes and environments consisting of high resolution implicit geometry using a minimal amount of memory. The fluid surface can also readily be represented as an out-of-core level set, possibly combined with the dual-resolution approach of Paper II.

3.4.4 Out-of-Core Particle Level Sets for Fluid Animation

As shown in figure 3.8 the PLS method uses significantly more memory than level set surface tracking alone when DT-Grids are used. Even though the particle compression method described in section 3.3.2 can reduce the memory footprint of the particles by close to 70% this still means that particle data uses well over an order of magnitude more primary storage than the level set itself. In order to reduce the amount of primary memory

Page-size	**32 pages**		**64 pages**		**128 pages**	
	LRU Demand	Opt Demand	LRU Demand	Opt Demand	LRU Demand	Opt Demand
0.5kB	0.6313	0.6450	0.6316	0.6606	0.6318	0.6911
1.0kB	0.6317	0.6600	0.6318	0.6906	0.6404	0.7490
2.0kB	0.6318	0.6896	0.6404	0.7483	0.6430	0.8573
4.0kB	0.6404	0.7469	0.6430	0.8564	0.8888	0.9439
	Our Demand	Our Prefetch	Our Demand	Our Prefetch	Our Demand	Our Prefetch
0.5kB	0.6421	0.9452	0.6580	0.9465	0.6888	0.9468
1.0kB	0.6546	0.9461	0.6858	0.9468	0.7454	0.9470
2.0kB	0.6794	0.9477	0.7406	0.9480	0.8533	0.9545
4.0kB	0.7305	0.9893	0.8482	0.9907	0.9436	0.9595

Table 3.1: Comparison of the page-hit-ratios of our page-replacement policy with prefetching disabled (*Our Demand*), prefetching enabled (*Our Prefetch*), LRU page-replacement without prefetching (*LRU Demand*) and the optimal page-replacement policy for a demand-paged system computed offline (*Opt Demand*) from a logged sequence of demand requests.

used by particle level sets as well as being able to represent PLS geometry at essentially arbitrary resolution we may employ our page-replacement and prefetching scheme to the PLS particles as well.

Similarly to the out-of-core level set it is however important for performance that the PLS particles are accessed as sequentially as possible. This can be ensured by making the particle storage order coincide with the storage order of the underlying DT-Grid level set. Our basic algorithm for ensuring this is presented in section 2.6.2 and a detailed description can be found in section 7.2 of Paper I.

In order to keep track of which particles belong to which DT-Grid cell we employ a mapping data structure that consists of a particle bin array and a particle array (see figure 2.7). Since one particle bin is associated with each DT-Grid cell, and the storage order is the same as the storage order of the DT-Grid, no explicit pointers to particle bins are needed. As can be seen in figure 2.7 each particle bin stores one index pointer into the particle array in addition to the number of particles contained in the array. This is done to allow for fast random access of the particles within each bin. However, if we assume that particle bins will always be accessed in sequential order this data structure can be made more compact by omitting the index pointer[14].

In order to achieve high data throughput[15] between primary and sec-

[14]The current index pointer can instead be calculated by each stencil point during sequential iteration through counting the particles in each bin.

[15]As measured in particles transferred per second

ondary storage our out-of-core particle streaming can also be combined with the fast particle compression algorithm described in section 3.3.2. Since the computational overhead of the particle compression scheme is negligible this effectively results in a 230% increase in particle throughput[16].

A performance benchmark of our out-of-core particle level set can be seen in table 3.2. This table is based on the results presented in table VII of Paper I and shows the relative processing time per grid-point for the advection of a DT-Grid PLS at different resolutions. The memory footprints of the different simulations can be found in table VI of Paper I. In table 3.2 the in-core, uncompressed performance of the smallest model is used as a reference. Note the slow performance of the 1024^3 resolution uncompressed in-core case. This is caused by the inefficient general-purpose paging of the operating system, a problem that is significantly reduced when our out-of-core scheme is used instead.

Two examples of fluid animations using our out-of-core PLS can be seen in figures 3.14 and 3.15. Note that the surface noise that can be seen on the sheets of fluid are artifacts of the PLS method trying to represent geometry that is becoming too thin to be accurately resolved on the DT-Grid. This issue along with the significant memory footprint of the PLS method led to the development of the dual resolution Eulerian surface tracking method and SAM/SP-SAM filters presented in chapter 4 and Paper II.

	800^3	1024^3	1500^3	4096^3
In-Core	1.00	10.30	NP	NP
Out-Of-Core	2.92	3.21	4.30	4.89
In-Core (Q)	1.04	1.05	1.10	NP
Out-Of-Core (Q)	1.06	1.13	1.14	1.71

Table 3.2: Benchmark showing the relative processing time per grid-point for the advection of a DT-Grid PLS at different resolutions. (Q) means that our quantized particle level set was used. In-core tests were Not Possible (NP) due to 32-bit hardware limitations for the 4096^3 simulation and only possible using quantization for the 1500^3 simulation.

3.4.5 Out-of-Core Streaming of Fluid Data

Our prefetching and page-replacement scheme can also efficiently be applied to all data fields used by our fluid solver (see section 2.6) with one exception: When solving the linear systems produced by the viscosity and pressure projection steps (see section 2.6.1) we store all data associated with the Conjugate Gradient [Shewchuk, 1994] method completely in-core. As can

[16]Each particle requires 70% less data to store, thus $\frac{1}{1-0.7} \approx 3.3$ times more particles can be transferred per second at a given data bandwidth.

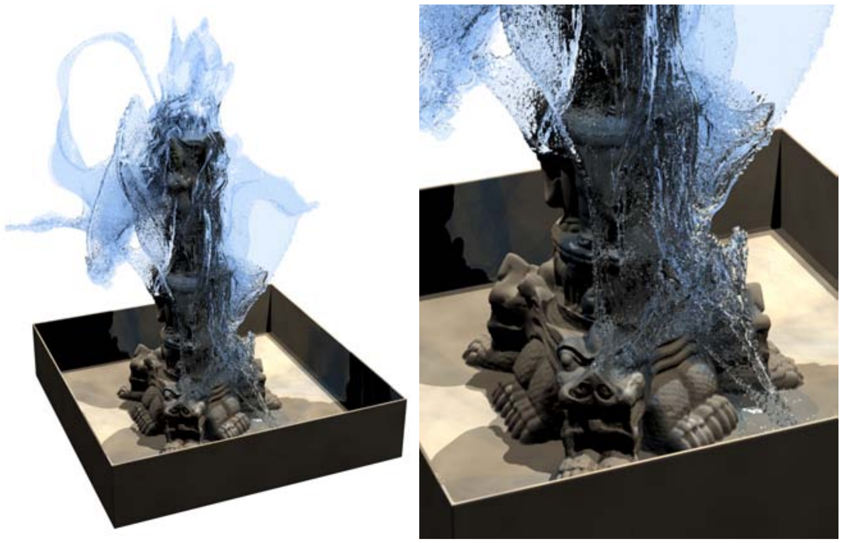


Figure 3.14: A fountain animation using our OOC and compression framework. OOC streaming and compression is used.

be seen in section 9.6 and Table X in Paper I the Conjugate Gradient method can be used with our OOC framework. However, this is a very data intensive algorithm and thus the performance penalty can be high, typically making OOC Conjugate Gradient inappropriate for fluid animation.

3.5 Summary

High resolution level set deformations and level set based free surface fluid animations can require a large amount of primary memory, especially if PLS surface tracking is used. In order to decrease the memory footprint associated with these methods Paper I and this chapter proposes the use of specialized on-line compression schemes and out-of-core streaming of data. The primary aim of our methods is the surface tracking aspect of free surface fluid animation, however our prefetching and page-replacement scheme is also applicable to the fluid solver itself.

Using our level set compression methods the size of a DT-Grid level set can be reduced by between 76 and 93 percent. However, the performance penalty is relatively high with level set computations taking up to 14 times longer than the uncompressed case. Although this is typically fast compared to general purpose compression methods it is slow compared to in-core speeds. The main reason for the reduced performance is however the compression of the level set distance values. Thus in order to maintain close to in-core performance when animating compressed level sets a faster

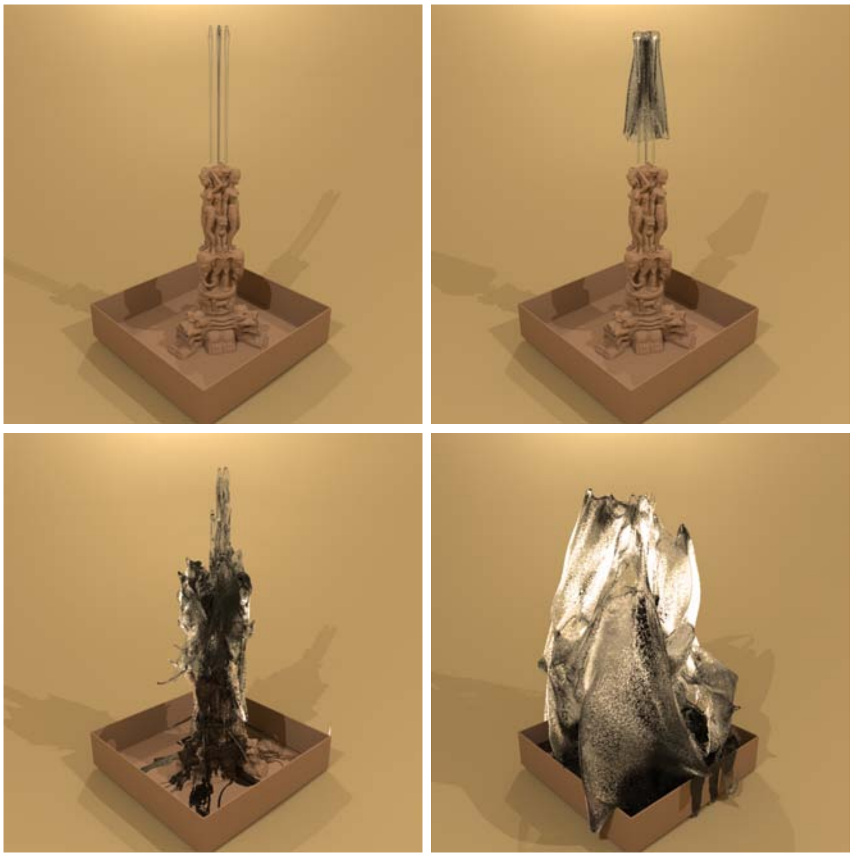


Figure 3.15: Frames 151, 351, 601 and 798 from a fluid animation of a large fountain. At its peak complexity the animation has an effective grid resolution of 600x1080x470 voxels, contains 580 million particles and 43 million voxels in the DT-Grid. The peak memory footprint of the simulation is 12.9GB. OOC streaming but no compression is used.

compression method for this data is desirable.

Our particle compression scheme can on the other hand reduce the size of a PLS by approximately 65 percent at a 5 percent performance penalty. The particle compression is also general purpose in the sense that it is applicable to any method where particles are stored in a grid.

Our prefetching and page-replacement scheme allows for very compact simulations, typically using only 128-256MB of primary memory. The page-replacement scheme is close to optimal for a demand paged system and often comes close to the ideal page-hit ratio of 1 when combined with prefetching.

The performance penalty of out-of-core fluid animation[17] depends heavily on which parts of the solver that uses out-of-core streaming. The data intensive Conjugate Gradient method becomes up to 14 times slower while the level set and compressed PLS can achieve close to in-core speeds[18]. The out-of-core PLS is however limited in the sense that performance is reduced if particles are allowed to move further than one grid cell per iteration of the advection algorithm. The reason for this is the large computational stencils necessary to cover all potential grid-cells into which a particle can move from its current position. For the fluid solver presented in section 2.6 this is however not an issue: The CFL stability condition related to explicit advection limits motion to less than the distance of a grid-cell per iteration.

When used for fluid animation our out-of-core and compression framework provides several methods for reducing or even virtually nullifying the memory requirements of each component of our fluid animation system. This allows for the ability to strike a compromise between the computational performance and the memory footprint of the simulation: Computationally expensive tasks as well as data that is not frequently used, for example solid geometry and surface velocity fields, can be stored using our out-of-core data structures. The memory demanding PLS can be compressed and if necessary stored out-of-core as well. This leaves significantly more memory for use when solving the Navier-Stokes equations. Potentially the entire fluid solver can be stored out-of-core, allowing for virtually limitless[19] resolution free surface fluid animations. The performance penalty for this can however be high if fast external storage devices are not used.

[17]Using the fluid solver presented in section 2.6.

[18]The OOC level set is 1.3 to 2 times slower compared to in-core performance while the OOC PLS using particle compression is between 1.06 and 1.71 times slower.

[19]The resolution is only limited by available secondary storage which can easily be in the order of Terabytes.

Chapter 4

Preserving Surface Details Without the Particle Pain

The focus of this chapter is the dual resolution method for Eulerian surface tracking presented in Paper II. The motivation behind this technique is presented in section 4.1. Next section 4.2 describes relevant previous work related to our method. Sections 4.3 and 4.4 then proceeds to present the main ideas behind Paper II focusing primarily on the geometric filters that together with DT-Grid level sets form the core of our method. This is followed by section 4.5 which presents a number of numerical experiments aimed towards validating the effectiveness of our method. This chapter is concluded in section 4.6 with a summary of our method.

4.1 Introduction

In Paper I the observation was made that the PLS method requires a large amount of memory compared to pure DT-Grid level sets. Typically this difference is close to two orders of magnitude as can be seen in for example figure 3.8 in chapter 3 as well as table 2 in Paper II. Although the particle compression scheme presented in Paper I will significantly reduce the memory footprint of the particles they still use more than an order of magnitude more memory than the DT-Grid level set itself[1]. As a result the resolution of a DT-Grid level set can be increased by at least a factor of three[2] or four (if no compression is used) before approaching the memory requirements of the PLS method.

Based on this observation we decided to develop a *dual-resolution* surface tracking approach for animating free surface fluids using the Stable Fluids method. The core idea behind our method is to replace the tracker particles of the PLS with a higher resolution level set. If comparable[3] results to the PLS method can be achieved before the memory footprint of the DT-Grid reaches that of the PLS method this approach can be seen as a memory efficient alternative to the PLS method.

[1]Assuming 70% compression.

[2]The memory requirement of a DT-Grid scales with the surface resolution. Thus doubling the resolution of a DT-Grid leads to approximately a factor four increase in memory requirement for the same geometry.

[3]Subjectively similar.

In order to allow level set surface tracking at a higher resolution than what is used to solve the Navier-Stokes equations, a connection must be established between the high resolution level set surface and the low resolution representation of this surface as seen by the Navier-Stokes solver. This can be achieved by downsampling the high resolution geometry onto the lower resolution Eulerian grid. If done naively this downsampling can however easily result in aliasing problems: Geometric features that are close to the size of, or smaller than, the low resolution sampling distance[4] risk being poorly represented or completely lost during the downsampling process. As a result entire regions of high resolution geometry may become invisible to the low resolution fluid solver which will thus be unable to properly animate this geometry.

In order to mitigate this type of aliasing problems we decided to develop a spatially adaptive geometric low-pass filter that is able to guarantee that all high resolution geometric features will, at least approximately, be preserved during downsampling. This property will ensure plausible animation of all parts of the high resolution geometry. The resulting method, including our Spatially Adaptive Morphology (SAM) geometric low-pass filter and its Sheet Preserving extension resulted in the publication of Paper II.

4.2 Related Work

Dual resolution surface tracking has previously been attempted for level sets by Goktekin et al. [2004] resulting in improved surface tracking at the expense of sampling related artifacts. Similarly Bargteil et al. [2006] encountered and discussed artifacts related to their use of a high resolution octree level set for their semi-Lagrangian contouring method. Recently Kim et al. [2009] coupled a high resolution particle level set to a lower resolution fluid solver in order to accurately track the effects of their vortex sheet method. Kim et al. also introduced a simple geometric filtering process they call the “Liquid Biased” (LB) filter in order to reduce the aliasing problems of their system.

The work of Kim et al. [2009] can be considered the most closely related to ours, however Kim et al. uses a particle level set at typically four times[5] the resolution of the underlying solver. This makes their method memory demanding which is not suitable for our goal of creating a fast and memory efficient Eulerian alternative to the PLS method. Furthermore, the Liquid Biased filter employed by Kim et al. is fairly simple and suffers from several weaknesses. Finally dual-resolution surface tracking has not previously been attempted using DT-Grids. Our use of these compact grids allow our method to be very memory efficient, thus potentially making it a compact

[4]I.e the size of a low resolution voxel.

[5]According to the figures presented in Kim et al. [2009].

Eulerian alternative to the PLS method.

As presented in Paper II our surface tracking method may also optionally employ sheet thickening in order to preserve thin sheets of fluid. Sheet thickening for this purpose has previously been proposed by Chentanez et al. [2007]. However, Chentanez et al. uses a particle based thickening method whereas our method is a purely Eulerian approach that is a natural extension of our SAM filter.

4.3 The Spatially Adaptive Morphology Filter

At the core of our dual resolution surface tracking approach is the concept of implicit geometry on two levels of resolution. Let φ_h denote the level set representation of the fluid interface on a high-resolution grid and let φ_l represent a filtered version of φ_h downsampled onto a lower-resolution grid. The Navier-Stokes equations are solved on the same grid as φ_l which results in the low resolution velocity field $\mathbf{v}_l$. Let Δx_l be the sampling distance used for φ_l and let Δx_h be the sampling distance used for φ_h. Let α denote the fraction $\frac{\Delta x_l}{\Delta x_h}$, i.e. the difference in grid resolutions given as a fraction. The challenge is now to derive φ_l from φ_h in a way that is feature-preserving and yet does not violate boundary conditions and introduce unnecessary topology changes. Once φ_l is defined we can compute $\mathbf{v}_l$ which in turn can be upsampled to $\mathbf{v}_h$ using for example simple trilinear interpolation. This finally allow us to advect φ_h through the level set equation

$$\frac{\partial \varphi_h}{\partial t} = \mathbf{v}_h \cdot \nabla \varphi_h \tag{4.1}$$

Simply defining φ_l from a naive down-sampling of φ_h is not appropriate since this can cause potentially severe aliasing problems [Bargteil et al., 2006; Kim et al., 2009]. Although φ_l is not intended to be explicitly rendered a poorly defined low-resolution interface can cause problems when solving the Navier-Stokes equations for $\mathbf{v}_l$ which in turn will affect φ_h through equation 4.1. This problem can be reduced by a uniform dilation of φ_h by $0.5\Delta x_l$ before downsampling. This is the approach of the LB filter employed by Kim et al. [2009]. The resulting surface will have no geometric features smaller than Δx_l and as a result all features in φ_h will be approximately represent in φ_l after downsampling. However, the global nature of the LB filter can cause several problems. First of all, if φ_h is in contact with a solid surface we may create a φ_l that penetrates this surface by as much as $0.5\Delta x_l$ potentially causing boundary problems. Furthermore, this dilation may cause topology changes in φ_l that are not present in φ_h. This includes premature merging (e.g. colliding droplets) and non-physical merging of, for example, separate surface details moving in parallel close to each other without actually touching. Finally, we observe that dilation

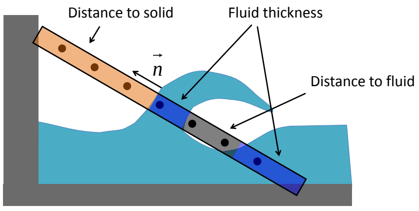


Figure 4.1: An illustration of a 7 point normal aligned stencil used by the SAM filter to estimate geometric thickness and distance.

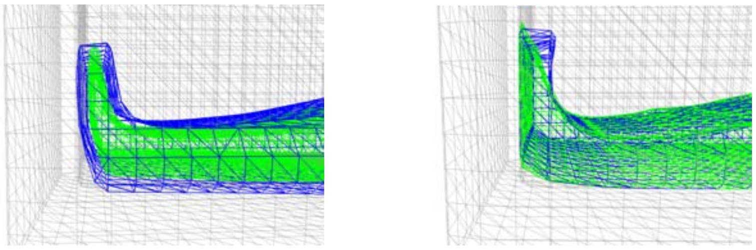
is not necessary for any features of φ_h already large enough to be resolved on φ_l.

These problems can be avoided by using a geometry aware filtering process, i.e. a filter that is aware of geometric feature thickness as well as any neighboring geometry. These ideas resulted in the creation of our Spatially Adaptive Morphology (SAM) filter. The basic idea behind this filter is to measure the feature thickness of different regions in φ_h as well as the distance to neighboring geometry. Our suggested method for computing these measurements is to use a normal-aligned computational stencil as depicted in figure 4.1. The distances between any implicit surfaces present in this stencil can then be estimated through linear interpolation of the level set distance field along the stencil. Based on these measurements adaptive dilation of φ_h can then be performed before creating φ_l through downsampling. The algorithm used to compute this dilation based on the distance estimates is presented in section 3 of Paper II.

Two examples of how the behavior of our SAM filter differs from the LB filter can be seen in figures 4.2 and 4.3. In both these figures the level set surface has been triangulated using the Marching Cubes [Lorenson and Cline, 1982] algorithm and rendered as a wireframe mesh where φ_h is colored green and φ_l is colored blue. The solid boundary is colored gray. Notice how the uniform dilation of the LB filter causes boundary penetration of in figure 4.2 and premature merging in figure 4.3. The SAM filter reduces these problems while still creating a plausible low resolution representation of the thin sheet in figure 4.2.

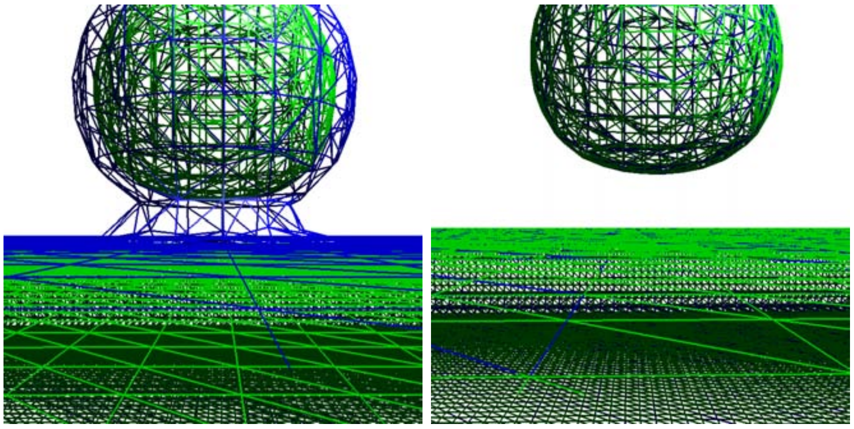
4.4 The Sheet Preserving SAM Filter

Since φ_h is represented on a uniformly sampled DT-Grid the situation may arise where parts of the fluid become too small or thin to be represented also by this high resolution level set. From a visual quality standpoint the most apparent problem caused by this scenario is the dissipation of sheets of fluid, potentially making large portions of fluid disappear into thin air. An example of such volume loss due to failed surface tracking can



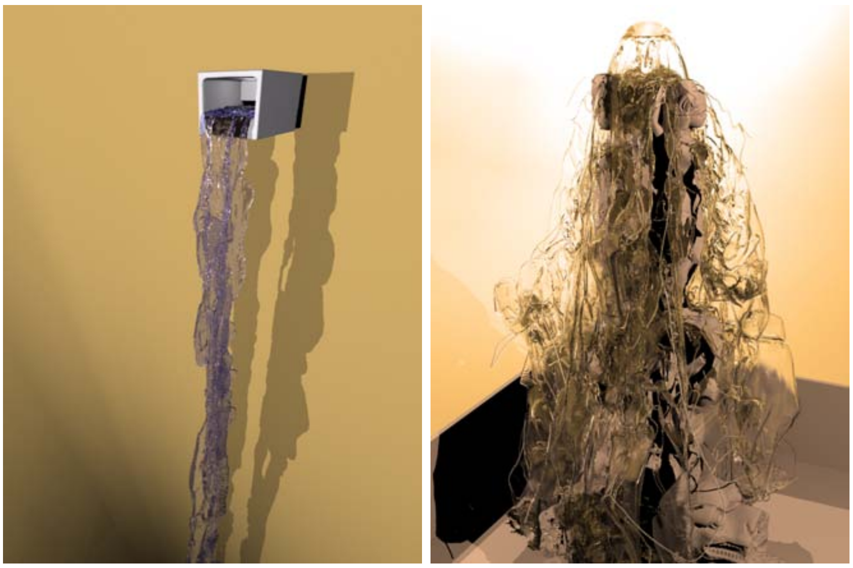
(a) (b)

Figure 4.2: An example showing the differences in behavior between the LB filter (figure 4.2(a)) and the SAM filter (figure 4.2(b)) close to solid boundaries. Here φ_h is colored green and φ_l is colored blue.



(a) (b)

Figure 4.3: An example showing the differences in behavior between the LB filter (figure 4.3(a)) and the SAM filter (figure 4.3(b)) close to another part of the fluid. Here φ_h is colored green and φ_l is colored blue.



(a) (b)

Figure 4.4: Two examples of fluid animations where the SP-SAM filter can be useful. Figure 4.4(a) shows a simple animation of a waterfall and figure 4.4(b) shows a massive, 20 meter tall fountain in the form of a statue.

be seen in figure 2.1. In scenarios where the preservation of thin sheets of fluid is paramount the adaptive and geometry aware nature of the SAM filter allows for a simple extension we call the Sheet Preserving Spatially Adaptive Morphology (SP-SAM) filter. The idea behind this filter is to use the feature thickness estimation of the SAM filter in order to define a dilation of φ_h as well as φ_l. This dilation is such that the thickness of a fluid sheet is always kept slightly above the width of a high resolution grid cell, thus guaranteeing the presence of the sheet at all times. This approach adds mass to the simulation and is consequently the most useful for scenarios where the preservation of mass is of lesser importance to visual quality than the existence of fluid sheets. Examples of this include scenes with large sources present, thus effectively masking small deviations in the fluid volume, and scenes where the entire body of fluid is not visible. The waterfall animation shown in figure 4.4(a) as well as the fountain shown in figure 4.4(b) are both good example of such scenarios. The SP-SAM filter can be realized by only slightly modifying the SAM filtering process as can be seen by the detailed description provided in section 5 of Paper II.

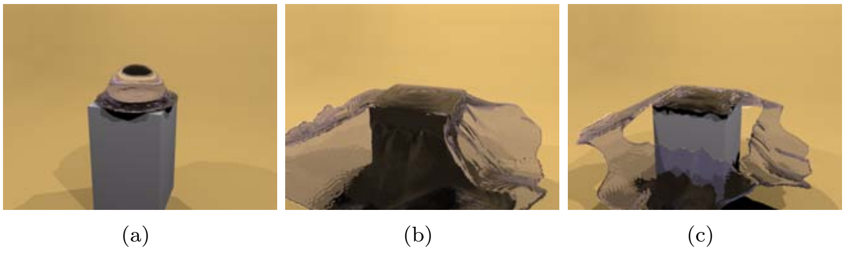


Figure 4.5: Examples of a problematic region when using the SP-SAM filter. Figure 4.5(a) shows a fluid sphere hitting the top of a pedestal. The resulting splash should at some point break up around the pedestal. If additional measures are not taken the SP-SAM filter produces the unphysical behavior shown in figure 4.5(b). By marking the pedestal as a solid that is allowed to break fluid sheets the expected behavior (figure 4.5(c)) can be achieved.

4.4.1 Extensions to the SP-SAM Filter

The basic SP-SAM filter presented above can easily preserve sheets too well. Consider for example the scenario of a sphere of water hitting the top of a pedestal (figure 4.5). In this scenario the fluid will stretch as it hits the pedestal until a thin sheet of fluid is present on the top of the pedestal. At this point the fluid sheet is expected to break apart, however the SP-SAM filter sees a thin sheet of fluid and will, if no additional care is taken, preserve this sheet indefinitely. As a result fluid will continuously flow from the top of the pedestal. This and similar scenarios are however easily avoided by exploiting the geometry aware nature of the SP-SAM filter. The simplest solution is to prohibit the filter from widening portions of φ_h that are in contact with solid geometry. This allows the fluid to break apart as it contacts the top of the pedestal while still preserving fluid sheets in midair. The resulting method will however also allow sheets of fluid to dissipate when in contact with the floor beneath the pedestal which may be undesirable. Thus it can be useful to specify in detail which solid objects or regions of solids should be allowed to break fluid sheets and which should not.

4.5 Results

The goal of our dual-resolution surface tracking algorithm is to create a purely Eulerian surface tracking method that can provide similar results to a PLS level set using less memory. Thus our main focus is on accurate advection and a low memory footprint. Since the PLS method is relatively fast we also put emphasis on low computational complexity. For these reasons we will limit ourselves to $\alpha = 2$ for the validation of our method.

In the examples below we compare our method with several other surface tracking methods. These methods and the motivation for including them as comparisons are as follows:

Single level set, 1X resolution This method is provided as a baseline reference. Both the level set and the fluid are resolved on the lower resolution grid typically resulting in poor surface tracking behavior.

PLS level set, 1x resolution This is the method we are primarily interested in comparing our results against. The PLS method allows significantly improved surface tracking at the expense of large amounts of memory and potentially noisy geometry.

Dual level set and SAM, 2x resolution Our method using the SAM filter and a DT-Grid level set at 2x the resolution of the grid on which the Navier-Stokes equations are solved. This method should preferably provide similar or better quality results than the PLS method.

Dual level set and SP-SAM, 2x resolution Our method using the SP-SAM filter and a DT-Grid level set at 2x the resolution of the grid on which the Navier-Stokes equations are solved. For this method we expect identical results to the SAM filter when no thin sheets are present in the fluid, while regions with sheets should be prevented from dissipating.

Single level set, 2x resolution This method is provided as a high resolution reference with both the level set and the Navier-Stokes equations solved on the higher resolution grid.

In order to achieve accurate advection of our higher resolution surface we solve equation (4.1) through high order explicit integration. For all examples the HJ WENO [Liu et al., 1994] and TVD Runge-Kutta [Shu and Osher, 1988] methods have been employed when advecting the fluid surface.

4.5.1 Breaking Dam Simulation

This example is intended to show the additional level of detail that can be obtained by employing our method. A comparison between our surface tracking approach and several other methods are shown in figure 4.6. Most notably we see that our dual interface approach provides comparative results to the PLS method without any surface noise (see figure 4.7). We also note that the results of the SAM and SP-SAM filters are essentially identical. This is to be expected since the SP-SAM filter should only widen surfaces when needed and thus leave the simulation untouched in well behaved cases such as this.

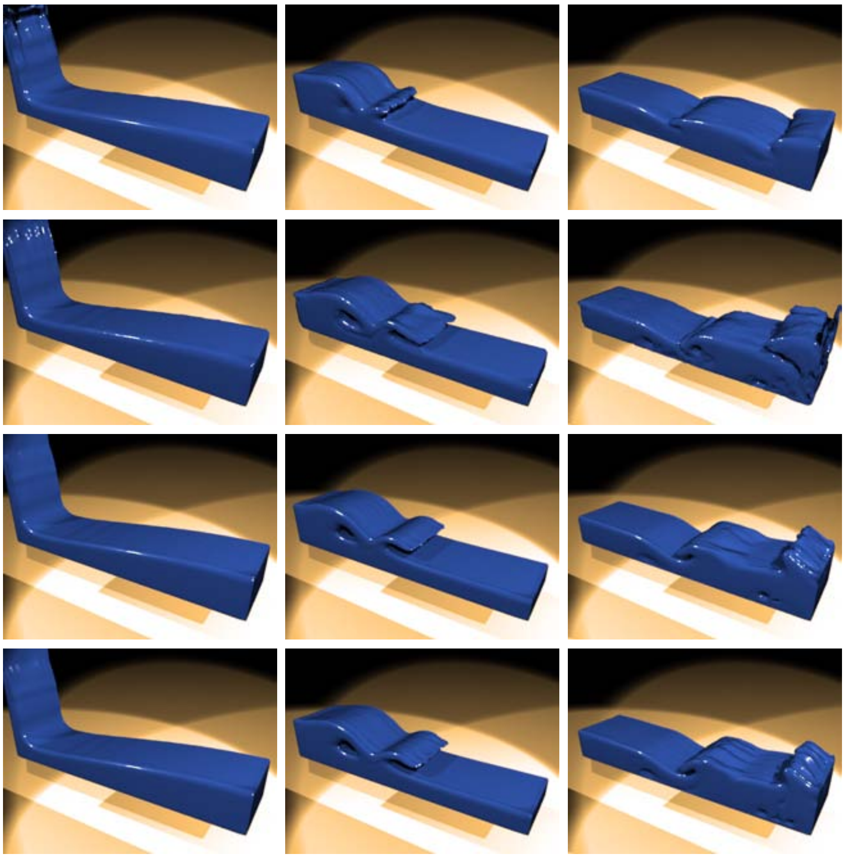


Figure 4.6: Each row of images above show three snapshots (frame 40, 73 and 110) of the breaking dam simulation using different surface tracking methods. From top to bottom these methods are: Single level set at 100^3 resolution, PLS at 100^3, Dual level sets and SAM filter at $100^3/200^3$. Dual level sets and SP-SAM filter at $100^3/200^3$.

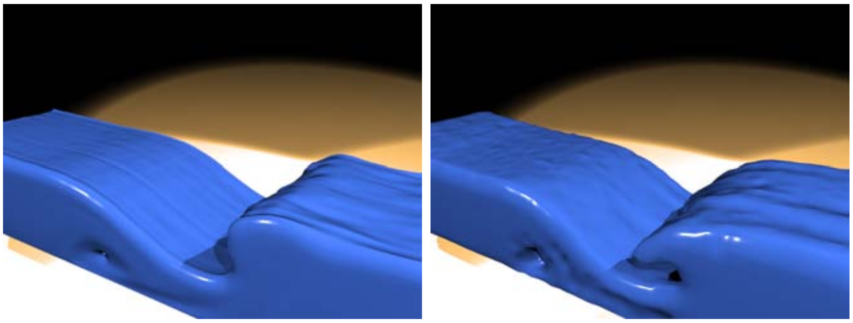


Figure 4.7: Comparison of the surface quality between our dual-resolution approach (left) and the PLS method (right). Here the surface noise that can appear when using the PLS method is clearly visible.

4.5.2 Splash Simulation

This example tests the behavior of our method during extreme stretching of the fluid. A comparison of frames from this simulation using different surface tracking methods can be found in figure 4.8. In this example we see the main advantage of the SP-SAM filter. Where both a high resolution single level set and a PLS fail to keep the sheets of fluid, our method kicks in and creates a smooth, noiseless surface. Note that as expected the surface still breaks apart on impact with the pedestal.

4.5.3 Flooding Simulation

This example is intended to show the behavior of our dual-resolution surface tracking method for a complex, visual effects oriented scenario. In this scenario a large amount of fluid is released in two small rooms, each at the end of a corridor. As a result the fluid will flood the corridors and continue into a central room, creating a fairly violent scenario with complex interface dynamics, including many sharp features and fluid sheets. An illustration of this scenario is presented in figure 4.9. A comparison of frames from this simulation using different surface tracking methods is provided in figure 4.10.

For this scenario the benefits of our method compared to both single level set surface tracking and the PLS method are fairly apparent. The extreme surface deformations and frequent fluid sheets makes the single level set fail utterly resulting in an almost complete loss of fluid volume by the time the fluid reaches the central room. The PLS method performs better, however, the limited spatial resolution still results in loss of thin features in spite of the particle correction. Furthermore noise caused by the particle representation can be seen in several places where the fluid surface

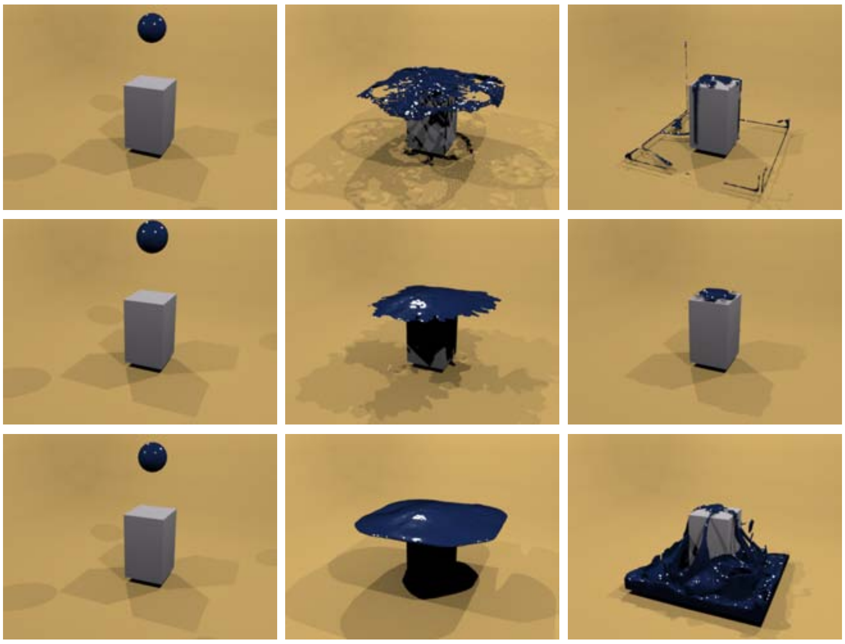


Figure 4.8: Each row of images above show three snapshots (frames 2, 30 and 69) of the Splash simulation using different surface tracking methods. From top to bottom the methods are: PLS at 100^3 resolution, dual level sets & SAM at $100^3/200^3$ and dual level sets and SP-SAM at $100^3/200^3$.

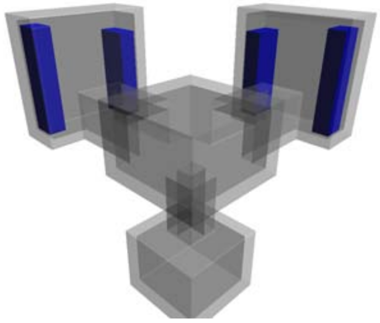


Figure 4.9: An illustration of the initial condition for the flooding simulation. The initial fluid volume is colored blue.

has sharp and/or thin features. Our dual-resolution approach with the SAM filter is able to preserve high-frequency geometry well while still providing plausible animation, providing significantly improved results compared to the unaided single level set. When combined with the SP-SAM filter sheets of fluid are also kept intact providing a result that is arguably more visually pleasing.

4.5.4 Computational Cost and Memory Footprint

Table 4.1 shows the average computation time per frame for the first couple of frames of the breaking dam simulation (see section 4.5.1) using different methods and resolutions. Table 4.2 shows the corresponding initial memory footprint associated with representing the fluid surface geometry. All methods use the DT-Grid data structure. The memory footprints for the SAM and SP-SAM methods represent the amount of memory required for representing the high resolution level set. In practice it may be desirable to store both the high and low resolution surfaces which increases the memory footprint of our method slightly[6].

As can be seen our method is slower than the PLS, but requires more than an order of magnitude less memory while still being able to provide significantly improved surface tracking performance compared to a single level set as is shown by the previously presented examples. Also note that significant increase in computation time when both the level set and the fluid solver are running at 2x resolution (last row in table 4.1). This is due to the fact that the computational complexity of the fluid solver scales with fluid volume while the surface tracking component scales with fluid surface area. Thus our method can achieve significantly improved surface tracking performance at a low computational cost compared to running the entire simulation at the higher resolution.

[6]Approximatly by the size of the single low resolution level set

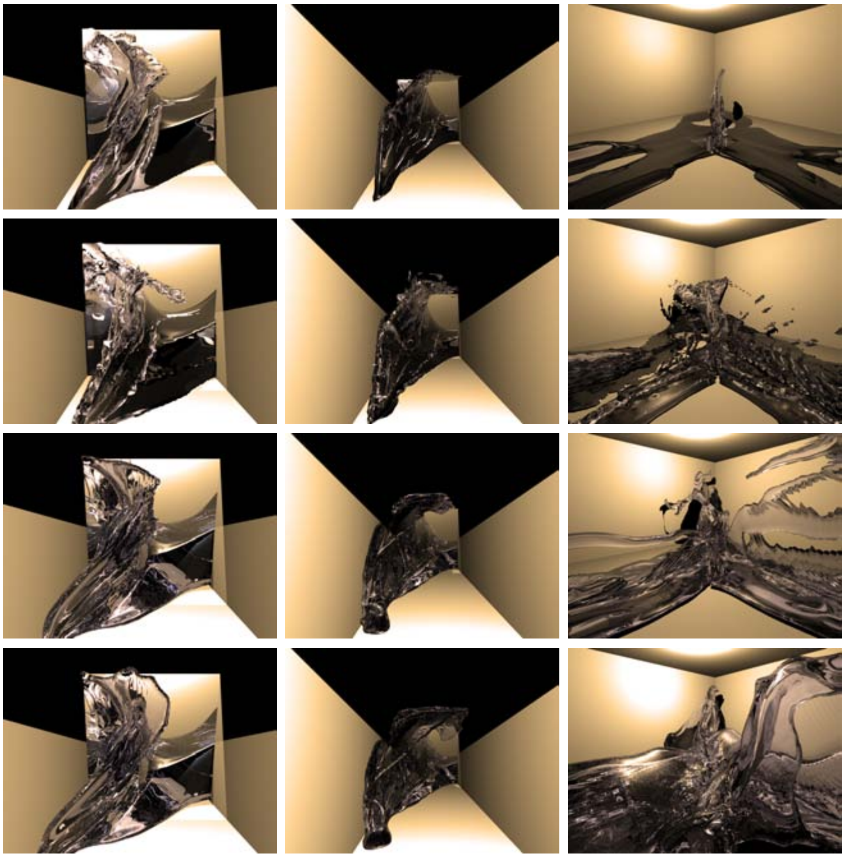


Figure 4.10: Each row of images above show three snapshots (frames 103, 134 and 245) from a simulation showing the flooding of a small building. From top to bottom the methods are: Single level set at 256^3 resolution, PLS at 256^3, dual level sets and SAM at $512^3/256^3$ and dual level set and SP-SAM at $512^3/256^3$.The first two images on each row show the initial flooding of one of the corridors while the third frame shows a view of the fluid entering the central room.

	100^3	200^3	300^3	400^3
Single level set	1.9	12.1	45.6	153
PLS	2.8	20.1	67.7	181
Dual + SAM	6.2	28.8	78.8	207
Dual + SP-SAM	6.3	31.8	96.6	212
Single level set 2x res	12.1	153	689	2261

Table 4.1: Benchmark results showing average time in seconds per frame for the initial frames of the Braking Dam simulation.

	100^3	200^3	300^3	400^3
Single level set	0.2	0.8	1.8	3.2
SAM & SP-SAM	0.9	3.6	8.2	15
PLS	15 (4.4)	60 (18)	136 (41)	242 (72)

Table 4.2: Table showing initial memory footprints (in MB) for the fluid surface of the breaking dam simulation. The SAM and SP-SAM results show the memory footprint of the high resolution surface. The PLS uses a particle density of 64. The numbers in parenthesis is the PLS method with 70% particle compression as can be achieved through the method presented in Paper I.

4.6 Summary

The main purpose of the work presented in Paper II and this chapter was to investigate dual-resolution level set surface tracking in order to achieve a memory efficient Eulerian alternative to the PLS method. The core components of our approach are the DT-Grid data structure and the SAM and SP-SAM filters allowing for plausible animation to be computed for all parts of the high resolution surface. Based on the results presented in section 4.5 and Paper II we consider our method successful. Our approach can provide significantly improved surface tracking compared to a single unaided level set, often achieving results that are subjectively close to or even better than the PLS method using an order of magnitude less memory than the PLS. Our method is also both robust and reliable, introducing no randomness[7] into the simulation and no parameters that need to be tweaked in order to maximize performance for a particular scenario.

The main weakness of our method compared to a PLS that it relies on Eulerian advection. Although we use high order numerical advection methods our level of accuracy cannot match the advection of Lagrangian particles, especially in high frequency regions like edges and corners.

[7]This can be the case with the PLS when random particle seeding is used.

Furthermore, the difference in resolution between $\mathbf{v}_l$ and φ_h can potentially lead to parts of φ_h moving in an unphysical manner. We have only experienced small artifacts of this type for $\alpha = 2$, however this can still be a problem.

Finally the sheet thickening of the SP-SAM filter adds mass to the simulation. However, the thickness of the enforced sheets are in the order of Δx_h, thus typically only covering a fraction of the low resolution grid cell. Consequently the amount of mass gained can potentially be reduced by more accurately estimating the actual part of a low resolution grid cell that contains fluid as described by φ_h. For this purpose volume fractions [Batty et al., 2007; Bridson, 2008] can potentially prove useful.

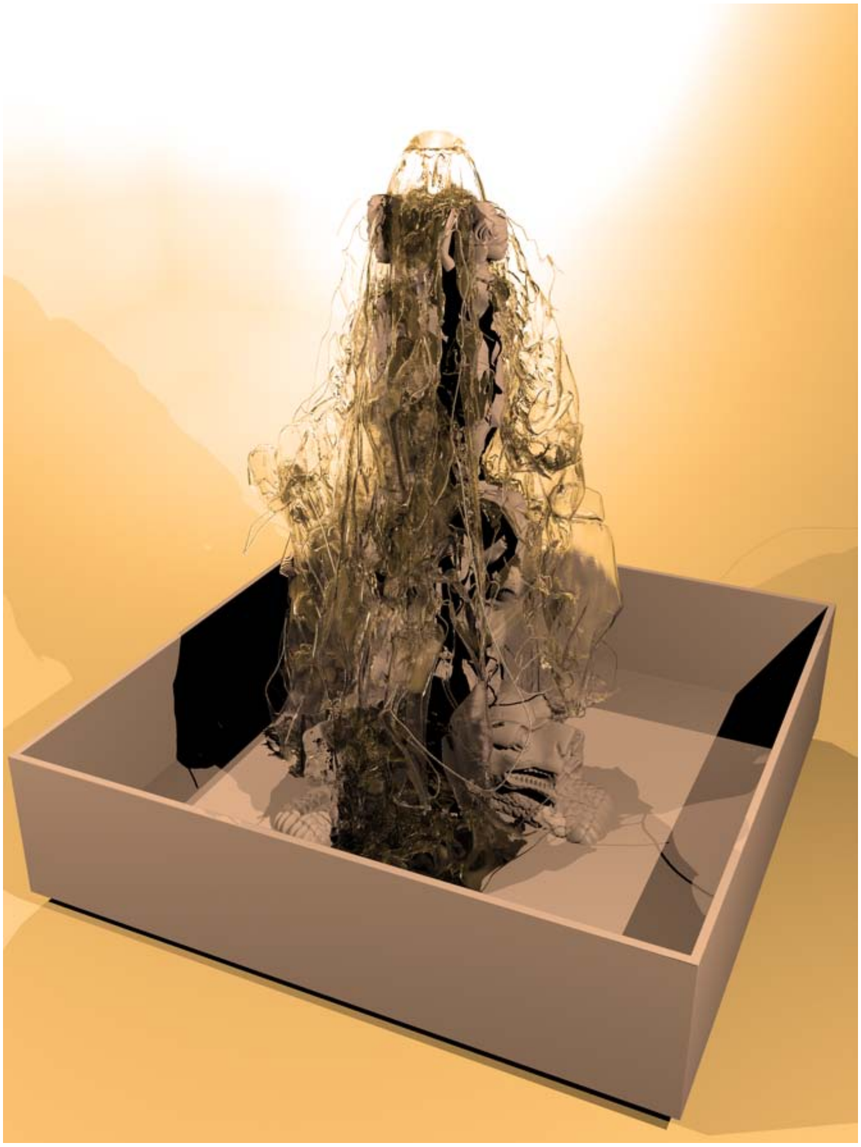


Figure 4.11: A high resolution rendering of a frame from a fountain animation using our dual resolution level set surface tracking with the SP-SAM filter (see chapter 4).

Chapter 5

An Ocean in a Box - Non-Reflecting Boundaries for Free Surface Fluid Animation

This chapter focuses on the non-reflecting boundary conditions for incompressible free surface fluid animation presented in Paper III. The chapter starts with an introduction to the problem that motivated the development of our boundaries. Following section 5.1 a brief background on non-reflecting boundaries, primarily focusing on the Perfectly Matched Layer (PML) method, is provided in section 5.2. Next section 5.3 explains how to define boundary regions and section 5.4 discusses the methods used in order to demonstrate the effectiveness of our boundaries. Sections 5.5 and 5.6 cover the introduction, design and evaluation of two fairly simple types of non-reflecting boundaries: the explicit and implicit dampening methods. This is then followed by section 5.7 presenting a method for deriving and solving the equations behind PML based wave absorbing boundaries for incompressible free surface fluid animation. Section 5.7 ends with a comparative evaluation of all three boundary types presented in this chapter. Section 5.8 then briefly describes how the PML boundaries can be implemented on MAC grids. This chapter is concluded in section 5.9 with a summary of our boundaries.

5.1 Introduction

When animating free surface fluids the situation may arises where one wishes to obtain a detailed simulation of a portion of a larger fluid volume. An example of this is capturing the complex dynamics of ocean waves interacting with a ship without simulating the whole ocean. In this situation it is tempting to create a smaller simulation domain focused only around a local area of interest, using solid wall boundaries to contain the fluid. The ocean outside this local simulation "box" can then potentially be animated using simpler and much faster methods[1] saving significant amounts of memory and computation time. A significant problem with this method is however that the presence of walls in the simulation agrees poorly with the

[1]Examples of such ocean models can be found in the work by Tessendorf [2004] and also in Bridsons book on fluid animation [Bridson, 2008].

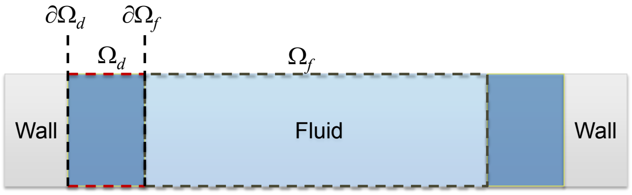


Figure 5.1: A 1D example of the different parts of the simulation domain involved when a non-reflecting boundary is employed.

actual scenario of a vast body of fluid. Waves approaching the edges of the local simulation will reflect off of the walls containing the fluid, typically returning towards the area of interest in the center of the simulation domain. These reflected waves, moving in the "wrong" direction, may easily break the illusion of a wider fluid domain that is intended.

In order to more accurately facilitate this type of local simulation approach a boundary condition is needed that allows waves to pass through unhindered while still containing the fixed volume of fluid present within the simulation domain. Our solution to this problem is to employ wave absorbing boundary regions close to the edges of the simulation domain. Within these regions alternate fluid equations can then be solved allowing waves to enter but absorbing them before they can reflect back into the unmodified simulation domain. An illustration of this idea can be seen in figure 5.1. Here the fluid domain Ω is divided into a regular fluid region Ω_f and a wave absorbing region Ω_d. Between Ω_f and Ω_d we have the internal boundary $\partial\Omega_f$ corresponding to the edge of the physical simulation domain, i.e. the domain which contains actual fluid. We also have the external boundary $\partial\Omega_d$, typically treated as a solid wall

In Paper III three different methods are presented for realizing this type of non-reflecting boundary condition for Eulerian animation of incompressible free surface fluids:

Explicit Dampening The explicit dampening boundary attempts to prevent wave reflection by explicitly modifying the velocity field of the fluid in Ω_d. The method is fast and robust but typically requires a large dampening region in order to be effective.

Implicit Dampening The implicit dampening boundary attempts to prevent wave reflection by adding a dampening term to the Navier-Stokes momentum equation in Ω_d. Accurately solving the resulting equations can require small timesteps in order to be stable and can thus be slow. However, when using the operator splitting approach of the Stable Fluids method, implicit integration can be applied resulting in an unconditionally stable solver. This results in a method that is very similar to explicit dampening. When integrated using small timesteps the implicit dampening method is more effective than explicit dampening. Relatively large dampening regions are however still required for this method to be effective.

Perfectly Matched Absorption Layer This approach aims to create a set of wave dampening equations that is specifically tailored to the Navier-Stokes equations. The resulting wave dampening equations are fairly complicated and introduces a number of additional variables. However, as is shown in Paper III, a Stable Fluids based solution algorithm can be constructed for these equations. Although the resulting scheme is not unconditionally stable the stability condition can still be made independent of the parameters of the boundary. The Perfectly Matched Layer approach results in highly effective boundaries that can be considered wave absorbing and not simply wave dampening. As a result this method can typically prevent wave reflection using a significantly smaller boundary region Ω_d than explicit and implicit dampening.

The above three methods for achieving reflection free boundaries have all been inspired by the success of equivalent methods in the fields of physics and engineering. However, to the extent of our knowledge the work presented in Paper III, and this chapter, is the first to derive and construct this type of boundary conditions for free surface fluid animation based on the incompressible Navier-Stokes equations.

5.2 Background

Undesired reflection from far-away, i.e. far-field boundaries has long been a problem in physics and engineering. In the field of computational aeroacoustics undesired reflection of pressure waves are for example often encountered when solving the compressible Navier-Stokes equations [Hu, 2008]. A fairly recent and promising method for preventing such wave reflections is the Perfectly Matched Layer (PML) approach. The idea behind a PML is to create a boundary condition that acts as a wave absorbing medium perfectly matched to the behavior of the governing equations of the problem being solved.

The first perfectly matched layer approach was introduced by Berenger in 1994 for computational electromagnetics [Berenger, 1994]. Berenger used a split-variable formulation that was later shown to be dynamically stable but only weakly well-posed [Abarbanel and Gottlieb, 1997]. Since then it has been shown that the PML method is equivalent to a complex change of variables in the frequency domain and that the PML equations can be formulated in unsplit physical variables [Chew and Weedon, 1994; Turkel and Yefet, 1998; Zhao and Cangellaris, 1996]. The PML method was first applied to computational fluid dynamics and computational aeroacoustics starting with the linearized Euler equations in [Hu, 1996]. In [Hu, 2001] it was later found that a necessary condition for the PML method to work in this scenario was that all physical waves have consistent phase and group velocities. This conclusion was also reached independently in [Bcache et al., 2003]. In response to this issue several new PML formulations have been developed. In [Hu, 2001] a stable PML method for the linearized Euler equations in the presence of a uniform mean-flow was proposed. Equivalent methods have also been presented in [Bécache et al., 2004; Hagstrom and Nazarov, 2002, 2003]. Recently the PML method has also been applied to the non-linear Euler [Hu, 2006] and Navier-Stokes equations [Hagstrom et al., 2005; Hu et al., 2008].

It is interesting to note that the fundamental idea of deriving absorbing boundary conditions by means of complex scaling of coordinates, which forms the very basis of the PML method, originates outside the field of computational fluid dynamics. While it is virtually impossible to track its exact origin, this idea of complex scaling has existed in quantum dynamics for a long time. Specifically, it has successfully been applied to the study of atomic and molecular dynamics in [Museth and Leforestier, 1996; Neuhasuer and Baer, 1989; Reinhardt, 1982] among others.

In the fields of computer graphics and fluid animation simple open boundaries[2] for incompressible gas and smoke simulation has been widely used for some time. Examples include [Fedkiw et al., 2001; Stam, 1999] among many others. Compressible gas and smoke simulation is not widely used in computer graphics, however PML boundaries concerning this problem have been well studied in, among others, the field of computational acoustics as presented above. Non-trivial, wave-absorbing open boundaries for the animation of incompressible free surface fluids was however first introduced in [Söderström and Museth, 2009] which constitutes preliminary results to Paper III.

A quick introduction to the field of non-reflecting boundaries and PMLs in the field of physics and engineering can be found in the evaluation provided by [Richards et al., 2004] and the progress review by Hu [2008] for computational aeroacoustics.

5.3 Constructing Dampening Regions

As presented in section 5.1 the non-reflecting boundaries presented in this thesis all rely on defining dampening regions in the simulation domain. Typically waves traveling into or out of these regions should be inhibited while waves traveling in parallel to the boundary can be left unaffected in order to avoid removing unnecessary momentum from the flow. For best numerical performance the amount of dampening should also be increased

[2]The inflow/outflow boundary conditions used for this type of simulations can be considered a type of open, although not reflection preventing, boundary condition.

smoothly the deeper into the boundary region a wave travels (see section 8.1 in Paper III). In order to efficiently define boundaries that allow for this behavior we presents two implicit methods: Simple boundaries based on cut-planes and a novel application of level set geometry for defining wave dampening boundaries of arbitrary shape and complexity.

5.3.1 Cut-plane Boundaries

Our first method uses a set of cut-planes to define $\partial\Omega_f$. The location of each cut-plane is defined by the position vector $\mathbf{p}_b$ and the orientation is given by the plane normal $\mathbf{n}_b$. We now make use of the well-known equation

$$\mathbf{n}_b \cdot \mathbf{x} + d = 0 \tag{5.1}$$

describing an implicit plane to which $\mathbf{n}_b$ is normal and d is a position constant that can be determined through $\mathbf{p}_b$. Provided that $\|\mathbf{n}_b\| = 1$ the shortest signed distance D between the implicit plane and a point $\mathbf{p}$ can readily be computed through the equation

$$D = (\mathbf{p} - \mathbf{p}_b) \cdot \mathbf{n}_b \tag{5.2}$$

This information can now readily be used to determine where in space the dampening schemes should be applied and to what extent. The normal of the closest plane can be used to estimate the amount of dampening applied in each spatial direction. An example of how the simulation domain can be partitioned using cut-plane boundaries is shown in figure 5.2. Note that for regions influenced by several cutplanes the closest distance must be computed by taking all relevant planes as well as edges and points into account. This method can also readily be extended for use with triangle mesh geometry, however for complicated meshes the closest distance computations can quickly become expensive.

5.3.2 Level Set Boundaries

The cut-plane method can easily be used for creating convex shapes. However, for concave boundary shapes and complex boundary geometry requiring large amounts of planes this method quickly becomes computationally expensive. In order to efficiently represent boundary shapes of arbitrary complexity we propose the use of DT-Grid level sets. A level set can readily encode a signed Euclidean distance field (see section 2.4.4). Thus classifying a point in space as either inside or outside Ω_d becomes a simple sign evaluation of the level set function φ. Likewise the closest distance from a point in space to the internal boundary $\partial\Omega_f$ can be readily obtained from φ at that point. Furthermore normals to the level set function can be estimated everywhere in space[3] using equation (2.40), thus making

[3]The normal will be harder to estimate around shocks of the level set distance function. Thus high order finite difference schemes may be needed.

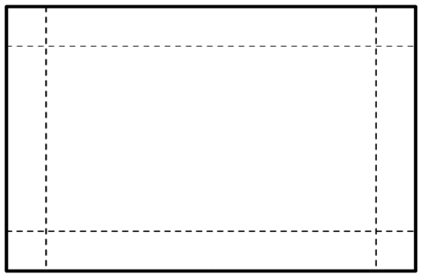


Figure 5.2: An example of how a rectangular simulation domain can be partitioned using cut-plane boundaries. The figure shows an ortographic projection of a 3D domain as seen from above with the dashed lines representing $\partial\Omega_f$ as defined by the cut-planes.

estimation of the amount of directional dampening that should be applied straight-forward as well. More information on our method for calculating directional dampening using level sets can be found in section 5.4 of Paper III.

5.4 Measuring Wave Absorption Efficiency

In order to demonstrate the basic performance of the three non-reflecting boundary conditions presented in this thesis a simple and controlled reference experiment has been constructed. The scenario is inspired by the effect of throwing a pebble into a large pond of still water: The pebble creates a splash that generates a ring-like wave pattern emanating from the point of impact. This wave pattern continues unbroken while energy is slowly lost due to the internal friction in the water until the surface of the pond is still yet again. In the actual simulation of this scenario a sphere of water is dropped in the middle a square basin generating ring-like waves that travel towards the edges of the simulation domain. In order to stay true to the envisioned scenario of a sphere of water dropped in the middle of a large pond, this ring-like wave pattern should continue uninterrupted until all the energy of the simulation has dissipated. However, in the reference case, solid walls are used at the edges of the simulation domain, causing wave reflections that in time produce a distinct interference pattern in the fluid domain. The performance of our non-reflecting boundaries can now be tested by investigating how much our boundaries are able to change this interference pattern towards the undisturbed ring-like wave pattern that is expected. In figure 5.3 three frames from this “Pebble in the Pond” reference simulation is shown.

As observed in Paper III the width of Ω_d is of importance for the

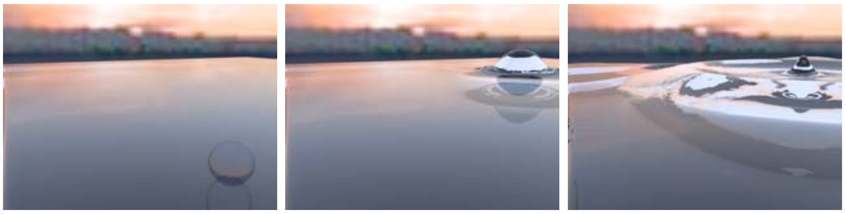


Figure 5.3: Rendering of three frames from the "Pebble in the Pond" reference simulation.

effectiveness of the non-reflecting boundaries presented in this chapter. For all wave absorption tests and graphs shown in this chapter, unless stated otherwise, the resolution of the simulation domain is 165x70x165 grid-points and the boundary width is 10 grid-points (i.e. 6% of the width of the simulation domain). These fairly small boundary regions are intended to stress the non-reflecting boundaries making differences in their effectiveness more apparent.

5.4.1 Fluid Surface Offset Visualizations

In order to visualize the wave patterns that arise during our experiments surface offset visualizations are used. These visualizations show the simulation domain from above using orthographic projection. The fluid surface is then color-mapped in order to display the offset of each surface point in comparison to the initial depth of the fluid. In scenarios like the pebble simulation depicted in figure 5.3, the initial position of the surface of the "pond" is used as a reference. An example of this type of visualization is provided in figure 5.4, showing a visualization of the example presented in figure 5.3. The visualization uses a color map where green represents the surface at the initial position, blue represents that the surface is below the original position and red represents that the surface is above the initial position. For each figure the color map is provided and the mapped value represents the surface height above a reference level[4].

5.4.2 Energy and Flux Graphs

In addition to the surface visualizations presented in section 5.4.1 two types of graphs are also used to show the effectiveness of our boundaries. The first type is a graph showing the total kinetic energy of the simulation. Since the non-reflecting boundaries presented in this thesis are all based on dampening of fluid motion to some degree, the effect of this dampening on the kinetic energy of the simulation as a whole is of interest. Ideally the

[4]As reference level we use the origin of the simulation domain

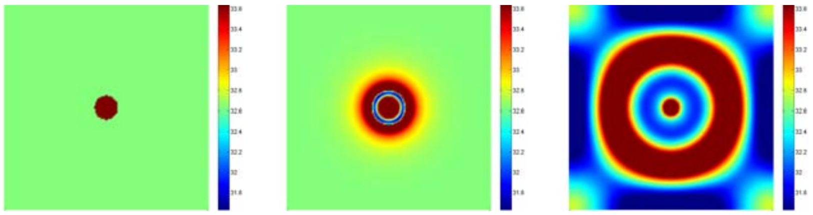
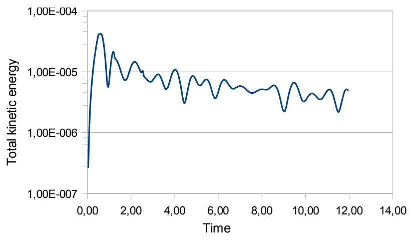
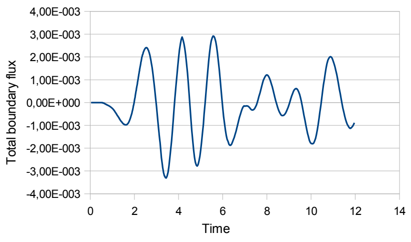


Figure 5.4: Example visualization of the surface offset in the "Pebble in the Pond" reference simulation. The three frames above are visualizations of the corresponding frames shown in figure 5.3.



(a)



(b)

Figure 5.5: Figure 5.5(a) shows a plot of the total kinetic energy of the "pebble in the pond" reference simulation depicted in figure 5.4. Figure 5.5(b) shows a plot of the total fluid flux through the inner boundaries of this simulation.

boundary should prevent wave reflection while preserving the kinetic energy as much as possible. An example of a kinetic energy graph for the "Pebble in the Pond" reference simulation is shown in figure 5.5(a). The kinetic energy of a simulation can vary drastically over time, thus a logarithmic scale is used for the energy axis for these graphs.

As surface waves pass through the internal boundary $\partial\Omega_f$ a net flux is temporarily generated through the boundary surface. In simple, highly symmetric scenarios like the "pebble in the pond" simulation this flux can be used as an indication of the amount of wave reflection present in the simulation domain. If no wave reflections are present the total flux through the boundaries should exhibit the behavior of a decaying oscillation. The reason for this is that each ring-like wave will pass through the boundaries in only one direction and with successively lower amplitude. However, if reflected waves are present this symmetry is broken and interference patterns will develop on the flux graph due to waves traveling in opposite directions. For these graphs positive flux is defined as flux from Ω_d to Ω_f. An example of a graph showing the total internal boundary flux in the

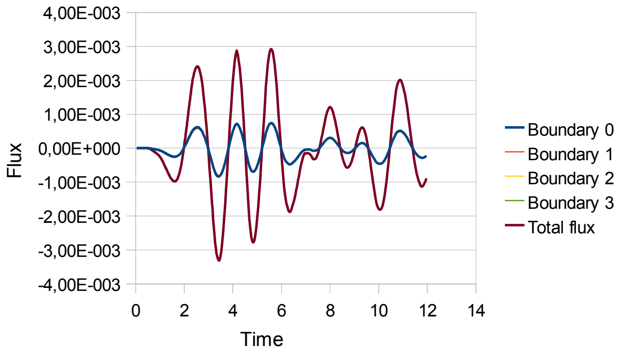


Figure 5.6: A graph sowing the total boundary flux through each cut-plane for the "pebble in the pond" reference simulation. The total flux from figure 5.5(b) is also shown.

"pebble in the pond" reference simulation is given in figure 5.5(b). In figure 5.6 the total flux through each of the four boundary planes is also shown, making the symmetry of the scenario apparent. In this simulation flux is measured through cut-planes positioned 10 grid cells away from each of the walls of the simulation domain (see figure 5.2 for an example of this), however no dampening is applied in Ω_d.

5.5 Explicit Dampening

The first and simplest type of non-reflecting boundary we propose for free surface fluid animation using the Stable Fluids method is the explicit dampening approach. This method relies on defining a target velocity $\mathbf{v}_{target}$ in Ω_d and then attempting to explicitly force the velocity of the fluid towards this target velocity. While the Navier-Stokes equations are solved the solution is regularly modified by enforcing the condition

$$\bar{\mathbf{v}} = \mathbf{v} - \sigma\left(\mathbf{v} - \mathbf{v}_{target}\right) \tag{5.3}$$

where σ is a dampening function, $\mathbf{v}$ is the solution to the undampened Navier-Stokes equations and $\bar{\mathbf{v}}$ is the solution after the explicit dampening. In order for this process to be effective equation (5.3) needs to be enforced frequently during the solution process, preferably at least once per iteration of the Stable Fluids algorithm. In Paper III it was found that a smooth and continuous σ function tend to yield better results in practice than steep and/or discontinuous functions. In [Richards et al., 2004] the dampening function

$$\sigma(x) = \sigma_{max}\left(1 + \frac{x - L}{L}\right)^{\beta} \tag{5.4}$$

which simplifies to

$$\sigma(x) = \sigma_{max}\left(\frac{x}{L}\right)^{\beta} \tag{5.5}$$

was used[5] for explicit boundaries in the context of computational aeroacoustics. Here L is the width of the dampening region and x is the distance from the internal boundary $\partial\Omega_f$. The maximum dampening σ_{max} and β are coefficients which determine the shape of the dampening function. In [Richards et al., 2004] it was found that the optimal value for these parameters is problem dependent and thus hard to estimate for a general scenario. However, according to figure 13 in [Richards et al., 2004] a $\beta \in [2, 2.5]$ yields good results. The target velocity $\mathbf{v}_{target}$ can be set to the mean-flow velocity of the fluid. Thus for many practical scenarios in visual effects $\mathbf{v}_{target} = 0$ is a good approximation. For $\sigma_{max} = 1$ this simply means that waves will lose all their energy by the time they reach $\partial\Omega_d$, i.e. all energy is dissipated over a distance of L. This is however not optimal since the goal with a non-reflecting boundary is for no reflected waves to leave the dampening region Ω_d. This means that waves can actually be allowed to retain a small amount of energy at $\partial\Omega_d$ as long as all their energy has dissipated by the time they reach $\partial\Omega_f$, i.e. after traversing a distance of $2L$. Finding a function for $\mathbf{v}_{target}$ that satisfies this condition is however non-trivial and during this project no literature was encountered that describe how to address this issue.

5.5.1 Explicit Dampening for the Stable Fluids Method

In order to apply explicit dampening boundaries this thesis proposes the following method: Before the pressure projection step (equations (2.45a) and (2.45b)) the explicit dampening condition (5.3) is applied inside the boundary region. In general the entire velocity vector can be dampened, however directional dampening may also be applied. For scenarios like waves moving over an ocean it is for example appropriate to only dampen the velocity components that are orthogonal to the equilibrium state of the fluid surface. Another approach is to apply the dampening in the normal direction of the boundary, a method that is inspired by the behavior of the PML boundary (see section 5.7). It can also be practical to apply the explicit dampening condition after the projection step. This will result in a

[5]The actual function in the paper is $\sigma(x) = \sigma_{max}\left(1 - \frac{x-L}{L}\right)^{\beta}$, however since it is also stated that $\sigma(x)$ is zero at the inner boundary ($x = 0$) the subtraction must be a typo.

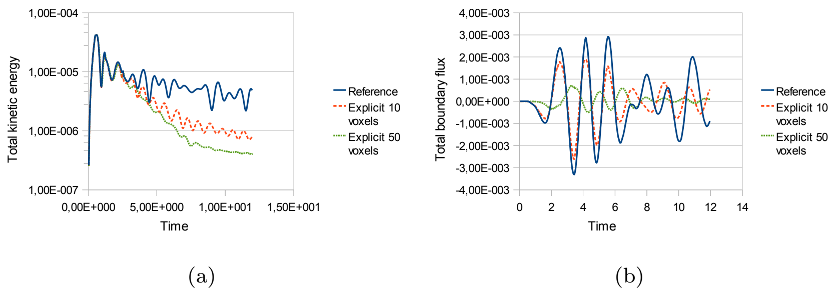


Figure 5.7: This figure shows a comparison between the undampened "pebble in the pond" reference simulation and the same simulation using explicit non-reflecting boundaries with Ω_d 10 and 50 grid-points wide. Figure 5.7(a) shows the kinetic energy of the simulation and 5.7(b) shows the boundary flux.

boundary where the fluid velocity field is temporarily divergent during the advection of the fluid surface. This in turn allows the explicit boundary to absorb mass as well as energy, potentially resulting in improved dampening performance at the cost of mass-loss. Note that only local mass-loss will occur since in this scenario the projection will be applied before explicit dampening during the next iteration of the Stable Fluids algorithm.

5.5.2 Evaluation

The main advantage of the explicit dampening method is that it allows unconditionally stable animation of free surface fluids with non-reflecting boundaries using the Stable Fluids approach. As a result fast and robust fluid animations can be obtained with reduced boundary reflections. However, as can be seen from the graphs in figure 5.7 and the wave pattern visualizations in figure 5.8 large dampening regions are required the method to be effective. Even then the large boundaries causes excessive amounts of kinetic energy to be removed from the simulation. This necessity to significantly increasing the effective size of the simulation domain also makes the method slow and not very memory efficient in practice[6]. For this evaluation equation (5.5) was used with $\beta = 2$, $\sigma_{max} = 1$ and $\mathbf{v}_{target} = 0$.

5.6 Implicit Dampening

The idea behind implicit dampening is to prevent wave reflection by including an additional velocity dampening term in the Navier-Stokes equations. Thus the dampening of velocity will be contributed by internal forces in-

[6]Assuming that low amounts of wave reflection is desired.

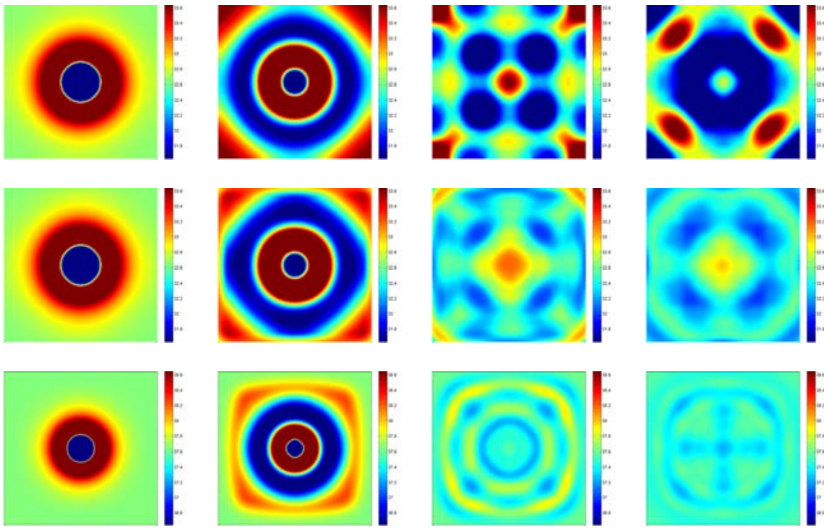


Figure 5.8: This figure shows a comparison between the wave patterns of the undampened "pebble in the pond" reference simulation (top row) and the same simulation using explicit non-reflecting boundaries with Ω_d 10 and 50 grid-points wide (middle and bottom rows respectively). The visualizations (left to right) correspond to times 1.1, 2.4, 7.2 and 10 seconds into the fluid animation.

stead of explicit external modification of the velocity field as is the case for explicit dampening. This internal force term, typically modeled as $\sigma\rho\mathbf{v}$ is added to the momentum equation (2.15), resulting in the dampened Navier-Stokes momentum equation

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) + \mathbf{v}\cdot\nabla(\rho\mathbf{v}) = -\nabla p + \nabla\cdot\mathbf{T} + \mathbf{f}_{ext} - \sigma\rho\mathbf{v} \tag{5.6}$$

where σ is a dampening function, for example equation (5.5). Due to the implicit nature of this boundary condition no finite value exists for the maximum dampening σ_{max} that will guarantee that all fluid flow has stopped at the outer boundary. Instead the only known property is that larger values for σ_{max} will result in larger forces counteracting fluid motion in the boundary region. This makes choosing an appropriately large σ_{max} a non-trivial and problem dependent issue. However, by including the dampening effect in the Navier-Stokes equations no explicit decision needs to be made about when or how often the boundary condition should be applied (as is the case for explicit dampening). Instead this will automatically be taken care of as part of the solution of the (dampened) Navier-Stokes equations.

5.6.1 Implicit Dampening for the Stable Fluids Method

Since implicit dampening acts as an additional internal force (or acceleration depending on the formulation) this effect can be integrated into a Stable Fluids based free surface fluid animation system quite easily. The operator-splitting approach allows the implicit dampening term to be solved as a separate step by integrating the equation

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) = -\sigma\rho\mathbf{v} \tag{5.7}$$

for example using the TVD Runge-Kutta time integration scheme. This will however result in a stability condition where the size of a stable timestep is proportional to σ and thus the resulting Stable Fluids method is not unconditionally stable. Stable integration can however be achieved by solving equation (5.7) through implicit integration. Using first order backward Euler time integration this results in the following update for the fluid velocity:

$$\mathbf{v}_{t+\Delta t} = \frac{1}{1+\Delta t\sigma}\mathbf{v}_t \tag{5.8}$$

which will be stable for any $\Delta t \geq 0$, $\sigma \geq 0$. However the accuracy of this method will be low if large timesteps Δt are taken. Indeed comparing equation (5.8) with equation (5.3) makes it clear that this approach is essentially explicit dampening with a somewhat more convoluted and timestep dependent dampening function. For the tests and results presented in this

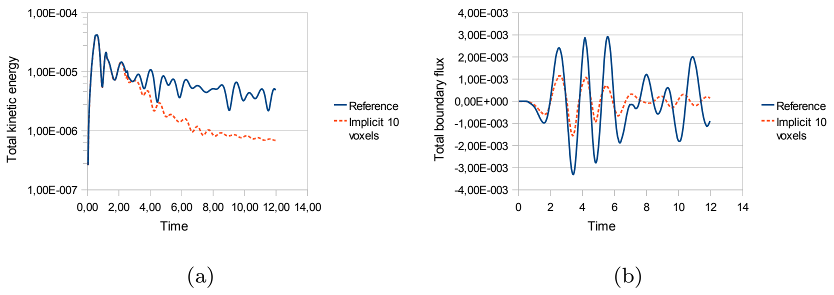


Figure 5.9: This figure shows a comparison between the undampened "pebble in the pond" reference simulation and the same simulation using implicit non-reflecting boundaries with Ω_d 10 grid-points wide. Figure 5.9(a) shows the kinetic energy of the simulation and 5.9(b) shows the boundary flux.

thesis equation (5.7) is integrated using explicit methods, thus making the method somewhat slow but also potentially more accurate.

5.6.2 Evaluation

The total energy and flux of implicit dampening as compared to the undampened "pebble in the pond" reference simulation can be seen in figure 5.9. The wave pattern comparisons are presented in figure 5.10. Equation (5.5) was used as dampening function with $\beta = 2$ and $\sigma_{max} = 77$. As can be seen the effectiveness of the implicit dampening condition is, when using explicit integration, somewhat improved over explicit dampening. Energy is conserved well in the initial stage of the simulation until waves enter the boundary regions. At this point the boundary condition removes the kinetic energy of the waves as they leave the simulation domain, leaving relatively small reflections returning from the boundaries. However, as can be seen in figure 5.10 relatively distinct interference patterns still develop. The cost of the increased effectiveness of the implicit dampening method is primarily the small time-steps necessary for accurate explicit integration of the velocity dampening term due to the large σ_{max} necessary when narrow boundaries are used.

5.7 Wave Absorbing Boundaries - The Perfectly Matched Layer Approach

As can be seen from the behavior of the relatively simple explicit and implicit wave dampening boundary conditions presented in sections 5.5 and 5.6, non-reflecting boundaries can be achieved through only slight modifications to the Stable Fluids algorithm. However both these methods

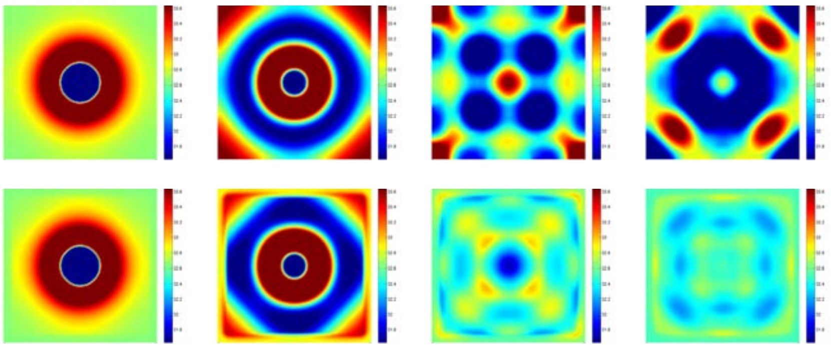


Figure 5.10: This figure shows a comparison between the wave patterns of the undampened "pebble in the pond" reference simulation (top row) and the same simulation using implicit non-reflecting boundaries (bottom row) with Ω_d 10 grid-points wide. The visualizations (left to right) correspond to times 1.1, 2.4, 7.2 and 10 seconds into the fluid animation.

tend to be slow if low boundary reflections are desired: Explicit dampening becomes slow since large boundary regions are needed in order to make the method effective. Implicit dampening can be effective using smaller boundaries, however this typically requires large values of σ_{max}, which in turn results in small time-steps in order to achieve an accurate dampening solution. Using implicit integration for the implicit dampening approach also shows that this method is essentially equivalent to explicit dampening.

In order to obtain a boundary condition that is more effective than explicit and implicit dampening a Perfectly Matched Layer approach can be used. A PML is a theoretical concept representing a perfectly wave absorbing medium and by constructing boundary regions consisting of this medium a perfect non-reflecting boundary can theoretically be obtained. Realizing a PML for a specific problem, like fluid flow described by the Navier-Stokes equations, is however a non-trivial task.

In this section we will derive a PML boundary condition for incompressible free surface fluid animation. Our derivation is based on previous work by Hu et al. [2008], showing how to obtain a PML boundary for the compressible Navier-Stokes equations without the presence of a free surface or external forces.

5.7.1 Obtaining a PML Boundary Condition for the Incompressible Navier-Stokes Equations

The core idea behind realizing a PML for the Navier-Stokes equations is the application of complex coordinate scaling. This coordinate transform is typically written as

$$x \rightarrow x + \frac{i}{\omega} \int_{x_0}^{x_1} \sigma(x) dx \tag{5.9}$$

where ω represents angular frequency and $\sigma(x)$ is a scaling function determining the amount of stretching. The variable x_0 represents the position of $\partial\Omega_f$, x_1 is the position of $\partial\Omega_d$ and i is the imaginary unit[7]. We will later see that $\sigma(x)$ is equivalent to the dampening function previously introduced in sections 5.5 and 5.6. Basic intuition for the behavior of equation (5.9) can be obtained by considering its effect on the complex wave component

$$e^{i(kx-\omega t)} \tag{5.10}$$

If (5.9) is applied to (5.10) this results in

$$e^{i(kx-\omega t)} e^{-\frac{k}{\omega} \int_{x_0}^{x_1} \sigma(x) dx} \tag{5.11}$$

As can be seen the second part of equation (5.11) represents exponential decay, assuming $\frac{k}{w} > 0$ and $\sigma(x) > 0$. Thus exponential dampening can be achieved for the amplitude of wave components traversing the region Ω_d where $\sigma(x) > 0$. Note that if $\frac{k}{w} < 0$ for $\sigma(x) > 0$ equation (5.11) instead represents an exponential increase of wave component amplitude. This scenario can occur when a mean-flow is present in the fluid and thus special care needs to be taken in such a scenario. Typically this involves offsetting the fluid velocity in order to remove the mean-flow as is described in [Hu et al., 2008] among others. Although we expect that this method can be applied to our PML boundaries the focus of Paper III and this chapter is PML boundaries for fluid flow where a mean-flow is not present and thus the issue described above will not be considered further.

Based on the transformation (5.9) the set of PML boundary equations for the incompressible Navier-Stokes equations can be derived. This set of equations will be the PML equivalent of equation (5.6). In order to keep the derivation compact the incompressible Navier-Stokes equations on conservation form, i.e.

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_1}{\partial x} + \frac{\partial \mathbf{f}_2}{\partial y} + \frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_{ext} \tag{5.12}$$

[7]The mathematical constant that satisfies $i^2 \equiv -1$

is used where

$$\mathbf{f}_1 = \begin{pmatrix} v_x \\ v_x^2 + \frac{p}{\rho} - \tau_{xx} \\ v_x v_y - \tau_{yx} \\ v_x v_z - \tau_{zx} \end{pmatrix}, \mathbf{f}_2 = \begin{pmatrix} v_y \\ v_x v_y - \tau_{xy} \\ v_y^2 + \frac{p}{\rho} - \tau_{yy} \\ v_y v_z - \tau_{zy} \end{pmatrix},$$

$$\mathbf{f}_3 = \begin{pmatrix} v_z \\ v_x v_z - \tau_{xz} \\ v_y v_z - \tau_{yz} \\ v_z^2 + \frac{p}{\rho} - \tau_{zz} \end{pmatrix} \mathbf{u} = \begin{pmatrix} 1 \\ v_x \\ v_y \\ v_z \end{pmatrix}, \mathbf{a}_{ext} = \begin{pmatrix} 0 \\ \frac{f_x}{\rho} \\ \frac{f_y}{\rho} \\ \frac{f_z}{\rho} \end{pmatrix}$$

and

$$\tau_{xx} = 2\nu \frac{\partial v_x}{\partial x}, \tau_{yy} = 2\nu \frac{\partial v_y}{\partial y}, \tau_{zz} = 2\nu \frac{\partial v_z}{\partial z},$$
$$\tau_{xy} = \nu \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right), \tau_{xz} = \nu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right),$$
$$\tau_{yz} = \nu \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right)$$

Here $\mathbf{a}_{ext}$ is the external force vector $\mathbf{f}_{ext}$ scaled by the constant density ρ, i.e. external acceleration. The above equations are essentially the compressible Navier-Stokes equations on conservation form, i.e. equation (2.19), where the energy equation has been dropped and the stress tensor $\mathbf{T}$ is somewhat simplified due to the fact that $\nabla \cdot \mathbf{v} = 0$ for an incompressible fluid. Note that if the assumption is made that ν is constant $\mathbf{T}$ can be simplified even further (see section 2.2.1) obtaining the relation

$$\mathbf{T} = \nu \nabla \mathbf{v} \tag{5.13}$$

Equation (5.12) is written on partial differential form and thus it is convenient to rewrite the integral equation (5.9) as its differential equivalent

$$\frac{\partial}{\partial x} \to \frac{1}{1 + i\frac{\sigma_x}{\omega}} \frac{\partial}{\partial x}, \quad \frac{\partial}{\partial y} \to \frac{1}{1 + i\frac{\sigma_y}{\omega}} \frac{\partial}{\partial y}, \quad \frac{\partial}{\partial z} \to \frac{1}{1 + i\frac{\sigma_z}{\omega}} \frac{\partial}{\partial z} \tag{5.14}$$

Here ω represents temporal frequency and $\sigma_{\{x,y,z\}}$ is a dampening function in the x, y and z directions respectively.

We now start our derivation by applying equation (5.14) to (5.12) in order to obtain

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{1 + i\frac{\sigma_x}{\omega}} \frac{\partial \mathbf{f}_1}{\partial x} + \frac{1}{1 + i\frac{\sigma_y}{\omega}} \frac{\partial \mathbf{f}_2}{\partial y} + \frac{1}{1 + i\frac{\sigma_z}{\omega}} \frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_{ext} \tag{5.15}$$

At this point we will make a key assumption. We assume that the solution $\mathbf{u}$ is radiating outwards from Ω_f as a superposition of planewaves, i.e.

$$\mathbf{u}(\mathbf{x}, t) = \Sigma_{\mathbf{k},\omega} \mathbf{w}_{\mathbf{k},\omega} e^{i(\mathbf{k}\cdot\mathbf{x} - \omega t)} \tag{5.16}$$

where ω is the angular frequency, $\mathbf{k}$ is the wave vector and $\mathbf{w}_{\mathbf{k},\omega}$ are the, assumed to be constant, wave amplitudes. Thus

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\partial}{\partial t} \Sigma_{\mathbf{k},\omega} \mathbf{w}_{\mathbf{k},\omega} e^{i(\mathbf{k}\cdot\mathbf{x} - \omega t)} = -i\omega \mathbf{u} \tag{5.17}$$

The assumption $\frac{\partial \mathbf{u}}{\partial t} = -i\omega\mathbf{u}$ is typically used when deriving PML boundaries for wave-type equations and is also implicitly made by Hu et al. [2008]. A brief discussion of equation (5.17) and its relevance for PML boundaries of the wave equation can be found in example [Johnson, 2007]. This subject is also briefly discussed by Hu [2005] as well as Hu [2008]. Under the assumption (5.17) equation (5.15) can now be written as

$$-i\omega\mathbf{u} + \frac{1}{1+i\frac{\sigma_x}{\omega}}\frac{\partial \mathbf{f}_1}{\partial x} + \frac{1}{1+i\frac{\sigma_y}{\omega}}\frac{\partial \mathbf{f}_2}{\partial y} + \frac{1}{1+i\frac{\sigma_z}{\omega}}\frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_{ext} \tag{5.18}$$

which essentially constitutes the PML boundary equations. We now wish to remove the explicit frequency dependence of equation (5.18). For this we will follow the split-variable method employed by Hu et al. among others. Thus we will split the variables $\mathbf{u}$ and $\mathbf{a}_{ext}$ into the six[8] auxiliary variables $\mathbf{q}_{\{x,y,z\}}$ and $\mathbf{a}_{\{x,y,z\}}$ such that

$$\mathbf{q}_x + \mathbf{q}_y + \mathbf{q}_z = \mathbf{u} \tag{5.19a}$$
$$\mathbf{a}_x + \mathbf{a}_y + \mathbf{a}_z = \mathbf{a}_{ext} \tag{5.19b}$$
$$-i\omega\mathbf{q}_x + \frac{1}{1+i\frac{\sigma_x}{\omega}}\frac{\partial \mathbf{f}_1}{\partial x} = \mathbf{a}_x \tag{5.19c}$$
$$-i\omega\mathbf{q}_y + \frac{1}{1+i\frac{\sigma_y}{\omega}}\frac{\partial \mathbf{f}_2}{\partial y} = \mathbf{a}_y \tag{5.19d}$$
$$-i\omega\mathbf{q}_z + \frac{1}{1+i\frac{\sigma_z}{\omega}}\frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_z \tag{5.19e}$$

Multiplying equations (5.19c), (5.19d) and (5.19e) with $\left(1+i\frac{\sigma_x}{\omega}\right)$, $\left(1+i\frac{\sigma_y}{\omega}\right)$ and $\left(1+i\frac{\sigma_z}{\omega}\right)$ respectively now results in the equations

$$(-i\omega + \sigma_x)\,\mathbf{q}_x + \frac{\partial \mathbf{f}_1}{\partial x} = \left(1+i\frac{\sigma_x}{\omega}\right)\mathbf{a}_x \tag{5.20a}$$
$$(-i\omega + \sigma_y)\,\mathbf{q}_y + \frac{\partial \mathbf{f}_2}{\partial y} = \left(1+i\frac{\sigma_y}{\omega}\right)\mathbf{a}_y \tag{5.20b}$$
$$(-i\omega + \sigma_z)\,\mathbf{q}_z + \frac{\partial \mathbf{f}_3}{\partial z} = \left(1+i\frac{\sigma_z}{\omega}\right)\mathbf{a}_z \tag{5.20c}$$

At this point we introduce the auxiliary variables

$$\frac{\partial \mathbf{h}_x}{\partial t} = \mathbf{a}_x \tag{5.21a}$$
$$\frac{\partial \mathbf{h}_y}{\partial t} = \mathbf{a}_y \tag{5.21b}$$
$$\frac{\partial \mathbf{h}_z}{\partial t} = \mathbf{a}_z \tag{5.21c}$$

[8] In the 3D case.

and make the assumption that the $\mathbf{a}_{\{x,y,z\}}$ vectors follow the time-propagation of equation (5.17). As a result the explicit frequency dependence of equations (5.20a)-(5.20c) can be removed by once again using the assumption (5.17) for all relevant quantities. As a result we obtain the equations

$$\frac{\partial \mathbf{q}_x}{\partial t} + \sigma_x \mathbf{q}_x + \frac{\partial \mathbf{f}_1}{\partial x} = \mathbf{a}_x + \sigma_x \mathbf{h}_x \tag{5.22a}$$
$$\frac{\partial \mathbf{q}_y}{\partial t} + \sigma_y \mathbf{q}_y + \frac{\partial \mathbf{f}_2}{\partial y} = \mathbf{a}_y + \sigma_y \mathbf{h}_y \tag{5.22b}$$
$$\frac{\partial \mathbf{q}_z}{\partial t} + \sigma_z \mathbf{q}_z + \frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_z + \sigma_z \mathbf{h}_z \tag{5.22c}$$

At this point our derivation is almost complete. However the flux vectors of equation (5.22c) also contains spatial derivatives as part of the components of the stress tensor. Consequently we need to apply equation (5.14) to these components. We can do this by introducing the auxiliary variables

$$\mathbf{e}_x = \frac{\partial \mathbf{v}}{\partial x} \tag{5.23a}$$
$$\mathbf{e}_y = \frac{\partial \mathbf{v}}{\partial y} \tag{5.23b}$$
$$\mathbf{e}_z = \frac{\partial \mathbf{v}}{\partial z} \tag{5.23c}$$

and applying the complex coordinate scaling (5.14) to equations (5.23a), (5.23b) and (5.23c), thus obtaining

$$\left(1 + i\frac{\sigma_x}{\omega}\right)\mathbf{e}_x = \frac{\partial \mathbf{v}}{\partial x} \tag{5.24a}$$
$$\left(1 + i\frac{\sigma_y}{\omega}\right)\mathbf{e}_y = \frac{\partial \mathbf{v}}{\partial y} \tag{5.24b}$$
$$\left(1 + i\frac{\sigma_z}{\omega}\right)\mathbf{e}_z = \frac{\partial \mathbf{v}}{\partial z} \tag{5.24c}$$

which, if the auxiliary variables

$$\frac{\partial \mathbf{r}_x}{\partial t} = \mathbf{e}_x \tag{5.25a}$$
$$\frac{\partial \mathbf{r}_y}{\partial t} = \mathbf{e}_y \tag{5.25b}$$
$$\frac{\partial \mathbf{r}_z}{\partial t} = \mathbf{e}_z \tag{5.25c}$$

are introduced can be written as

$$\frac{\partial \mathbf{r}_x}{\partial t} + \sigma_x \mathbf{r}_x = \frac{\partial \mathbf{v}}{\partial x} \tag{5.26a}$$

$$\frac{\partial \mathbf{r}_y}{\partial t} + \sigma_y \mathbf{r}_y = \frac{\partial \mathbf{v}}{\partial y} \tag{5.26b}$$

$$\frac{\partial \mathbf{r}_z}{\partial t} + \sigma_z \mathbf{r}_z = \frac{\partial \mathbf{v}}{\partial z} \tag{5.26c}$$

At this point we have finally obtained the governing equations describing the PML boundary medium for an incompressible fluid under the influence of essentially arbitrary[9] external forces:

$$\mathbf{u} = \mathbf{q}_x + \mathbf{q}_y + \mathbf{q}_z \tag{5.27a}$$

$$\mathbf{a}_{ext} = \mathbf{a}_x + \mathbf{a}_y + \mathbf{a}_z \tag{5.27b}$$

$$\frac{\partial \mathbf{q}_x}{\partial t} + \sigma_x \mathbf{q}_x + \frac{\partial \mathbf{f}_1}{\partial x} = \mathbf{a}_x + \sigma_x \mathbf{h}_x \tag{5.27c}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} + \sigma_y \mathbf{q}_y + \frac{\partial \mathbf{f}_2}{\partial y} = \mathbf{a}_y + \sigma_y \mathbf{h}_y \tag{5.27d}$$

$$\frac{\partial \mathbf{q}_z}{\partial t} + \sigma_z \mathbf{q}_z + \frac{\partial \mathbf{f}_3}{\partial z} = \mathbf{a}_z + \sigma_z \mathbf{h}_z \tag{5.27e}$$

$$\frac{\partial \mathbf{r}_x}{\partial t} + \sigma_x \mathbf{r}_x = \frac{\partial \mathbf{v}}{\partial x} \tag{5.27f}$$

$$\frac{\partial \mathbf{r}_y}{\partial t} + \sigma_y \mathbf{r}_y = \frac{\partial \mathbf{v}}{\partial y} \tag{5.27g}$$

$$\frac{\partial \mathbf{r}_z}{\partial t} + \sigma_z \mathbf{r}_z = \frac{\partial \mathbf{v}}{\partial z} \tag{5.27h}$$

The above set of equations has obviously added a significant amount of complexity to the original incompressible Navier-Stokes equations. However, a number of interesting observations can be made. First of all the *directional* dampening functions σ_x, σ_y and σ_z have been introduced. As a result the PML equations can dampen motion differently in different spatial directions. Related to this the state vector $\mathbf{u}$ has been split into the three vectors $\mathbf{q}_x$, $\mathbf{q}_y$ and $\mathbf{q}_z$. Thus the first component in $\mathbf{u}$, i.e. the constant density[10], has been split into a vector of possibly time-dependent quantities the sum of which is the traditional density. Likewise each component of the velocity vector field has been split into three components the sum of which is the traditional speed in each direction. Based on these observations and the anisotropy of the PML medium the conclusion can be drawn that the traditional scalar density has been split into a first order tensor density and the traditional vector velocity has been split into a velocity tensor. The

[9]These equations are based on the assumption that the time-propagation of the external force field adheres to equation (5.17).

[10]This can be surmised for example by comparing equation (2.19) and (5.12).

contraction of these tensors (i.e. equation (5.27a)) will then result in the traditional velocity and density. Likewise the external force vector can also be interpreted as having been split into a tensor force field.

Equation (5.27b) does not provide any explicit indication of how $\mathbf{a}_{ext}$ should be split into $\mathbf{a}_x$, $\mathbf{a}_y$ and $\mathbf{a}_z$. Indeed looking at the system of equations (5.27a) - (5.27h) the split seems to be possible to perform arbitrarily as long as equation (5.27b) is satisfied. Whether or not this is true is as of now an open question that requires further investigation and is in this thesis considered to be future work. However, at this point two observations can be made. First, the PML boundary equations (5.27a) - (5.27h) only needs to be solved *in the boundary region.* Since $\sigma_{\{x,y,z\}}$ will be zero outside these boundaries the PML equations revert to the traditional Navier-Stokes equations where external forces pose no problem. Since most interactions in a fluid simulation will take part some distance away from the non-reflecting boundaries only global external forces must be represented in the boundary region by necessity. The most important global external force is the force of gravity. This force belongs to the category of conservative forces, i.e. forces that can be described as the gradient of a scalar field. This property allows the PML equations to be significantly simplified and also provides a plausible approach for splitting the external force vector: Assuming that

$$\mathbf{f}_{ext} \equiv -\nabla U \tag{5.28}$$

for some scalar field U. This behavior is very similar to that of the fluid pressure field p. Thus the scalar field U can be interpreted as an offset to the fluid pressure and can thus naturally be included in the flux vectors of the Navier-Stokes equations. As a result the equations (5.27c) - (5.27e) simplify to

$$\frac{\partial \mathbf{q}_x}{\partial t} + \sigma_x \mathbf{q}_x + \frac{\partial\left(\mathbf{f}_1 + \mathbf{U}_x\right)}{\partial x} = 0 \tag{5.29a}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} + \sigma_y \mathbf{q}_y + \frac{\partial\left(\mathbf{f}_2 + \mathbf{U}_y\right)}{\partial y} = 0 \tag{5.29b}$$

$$\frac{\partial \mathbf{q}_z}{\partial t} + \sigma_z \mathbf{q}_z + \frac{\partial\left(\mathbf{f}_3 + \mathbf{U}_z\right)}{\partial z} = 0 \tag{5.29c}$$

where

$$\mathbf{U}_x = \begin{pmatrix} 0 \\ \frac{U}{\rho} \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{U}_y = \begin{pmatrix} 0 \\ 0 \\ \frac{U}{\rho} \\ 0 \end{pmatrix}, \quad \mathbf{U}_z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{U}{\rho} \end{pmatrix}$$

The resulting PML equations for an incompressible fluid with conservative external forces, i.e. equations (5.27a), (5.29a) - (5.29c) and (5.27f) - (5.27g) can now be written in the somewhat simpler and more familiar form

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_1}{\partial x} + \frac{\partial \mathbf{f}_2}{\partial y} + \frac{\partial \mathbf{f}_3}{\partial z} + \sigma_x \mathbf{q}_x + \sigma_y \mathbf{q}_y + \sigma_z \mathbf{q}_z = \mathbf{a}_{ext} \tag{5.30}$$

which can also be written as

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \mathbf{a}_{ext} + \nabla \cdot \mathbf{T} - \frac{\nabla p}{\rho} - \mathbf{A}\sigma_x \mathbf{q}_x - \mathbf{A}\sigma_y \mathbf{q}_y - \mathbf{A}\sigma_z \mathbf{q}_z \quad (5.31a)$$

$$\nabla \cdot \mathbf{v} = -\mathbf{B}\sigma_x \mathbf{q}_x - \mathbf{B}\sigma_y \mathbf{q}_y - \mathbf{B}\sigma_z \mathbf{q}_z \quad (5.31b)$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \quad (5.32)$$

Note that equation (5.31a) is constructed from equation (2.21a) and requires the assumption of constant viscosity, i.e. equation (5.13). Likewise equation (5.31b) is obtained from equation (2.21b).

At this point the full potential of the PML boundary becomes clear. Equation (5.31a) looks very similar to implicit dampening, however velocity dampening can be applied *directionally* and is proportional to the components of a velocity tensor. Furthermore, even though these are boundaries for the incompressible Navier-Stokes equations a number of terms has appeared on the right-hand side of equation (5.31b) indicating that the fluid can behave in a pseudo-compressible manner in the boundary. This is to be expected since a truly open boundary should allow mass to move in and out of the fluid domain as waves pass through the boundary. Thus we can consider our PML based boundary wave *absorbing* instead of simply wave *dampening* as is the case for the explicit and implicit boundaries presented in sections 5.5 and 5.6.

However, as can be seen the four equations (2.21a) - (2.21b) have become twelve equations for the $\mathbf{q}$ fields, nine equations for the $\mathbf{r}$ fields and nine equations for the $\mathbf{e}$ fields for a grand total[11] of 30 equations. Additionally there is the three equations in (5.27a) relating $\mathbf{q}_{\{x,y,z\}}$ and $\mathbf{v}$. This makes solving the PML equations a computationally more complex task than solving the regular incompressible Navier-Stokes equations. However, as is shown in section 5.7.6 and 5.7.7 these equations can be solved approximately. This will reduce the number of equations and unknowns to 12, thus significantly reducing the computational complexity and memory overhead of our PML method. Furthermore, the solver presented in section 5.7.6 is fairly similar[12] to the Stable Fluids method, showing that PML boundaries can be employed without much complexity added to one of the common solution methods used in computer graphics.

5.7.2 A Dirichlet Boundary Condition for the Velocity Tensor

The PML boundary method introduces a tensor velocity field to the fluid. Thus a plausible Dirichlet boundary conditions is needed for this velocity

[11] Assuming conservative forces in the boundary regions.

[12] Essentially the only difference is explicit self-advection which makes these boundaries easily usable with the solver presented in section 2.6.

tensor at solid boundaries. The Dirichlet boundary condition for the velocity field $\mathbf{v}$ at a solid boundary is given by equation (2.25). This simply states that the fluid velocity $\mathbf{v}$ along the boundary surface normal $\mathbf{n}_s$ should equal the velocity of the boundary $\mathbf{v}_s$ along the same normal. A plausible Dirichlet boundary condition for the velocity tensor $\mathbf{T}$ can be found by extrapolating this condition to the velocity related components of the tensor field, leading to the equation

$$\mathbf{n}_s^T \mathbf{T} = \mathbf{n}_s^T \mathbf{T}_s \tag{5.33}$$

Here $\mathbf{n}_s^T$ denotes the transposed solid surface normal. Equation (5.33) requires a tensor velocity field to be estimated for each solid boundary in the dampening region. However, since interactions generally take place outside this region, the only remaining boundary is that of the wall containing the fluid. Since this wall is typically stationary, $\mathbf{T}_s$ will in this case become the zero tensor, making equation (5.33) straight-forward to enforce.

5.7.3 PML Boundaries and the Free Surface

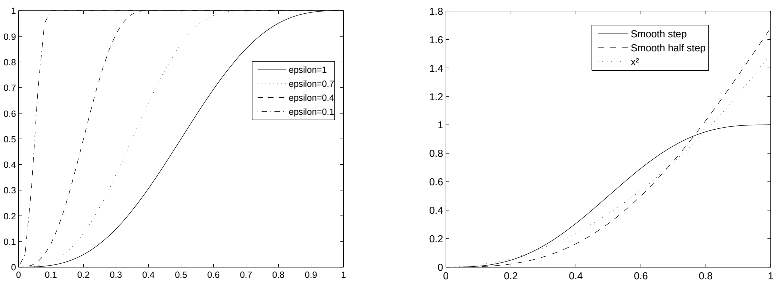
Due to the presence of a free surface an appropriate boundary condition for the velocity related $\mathbf{q}_{\{x,y,z\}}$ variables is needed at the fluid/air interface. For this a continuity boundary condition is used, i.e. it is assumed that the motion of the medium surrounding the fluid is completely governed by the motion of the fluid itself (see section 2.3.2). Thus the tensor equivalent of the velocity extension algorithm described in [Osher and Fedkiw, 2002] among others is applied to the $\mathbf{q}_{\{x,y,z\}}$ variables. This amounts to solving the equation

$$\frac{\partial q_{i,j}}{\partial \tau} + \mathbf{n} \cdot \nabla q_{i,j} = 0 \tag{5.34}$$

for all components $q_{i,j}$ of the vectors $\mathbf{q}_{\{x,y,z\}}$. In equation (5.34) $\mathbf{n}$ is the fluid surface normal and τ is the fictitious time used to propagate the solution to a steady state.

5.7.4 The Dampening Function σ

The transition from the regular simulation domain to the PML boundary region can be achieved in several ways. The simplest approach is to assume that the dampening function σ is a step function. However, this is problematic since equations (5.29a) - (5.29c) will lead to terms of the form $\frac{\partial \sigma_x \mathbf{r}_x}{\partial x}$ (see equation (5.43)) which makes the choice of a discontinuous σ function inappropriate. Furthermore a smooth transition is expected to provide better numerical results since this should reduce the amount of numerically induced reflection that can occur when the PML equations are discretized. For explicit and implicit dampening a simple polynomial transfer function (equation (5.5)) is used. However, the exponential nature of (5.11) leads to the suspicion that a function more related to exponential



(a) (b)

Figure 5.11: In figure 5.11(a) the "smooth step" dampening function is shown for different values of ϵ. Figure 5.11(b) shows a comparison between the different transfer functions described in equation (5.5), (5.35) and (5.36). For the step functions $\epsilon = 1.0$ has been used and the polynomial function uses $\gamma = 2$. All functions have been normalized such that $\int_0^1 \sigma(x)dx = 0.5$.

growth could provide better results. For this reason Paper III investigates two additional transfer functions. The first is the smooth step function

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ \sigma_{max} & x > \epsilon \\ \left(\frac{x}{\epsilon} - \frac{1}{2\pi}\sin\left(\frac{2\pi x}{\epsilon}\right)\right)\sigma_{max} & x \in [0, \epsilon] \end{cases} \tag{5.35}$$

with the maximum amplitude σ_{max} and width ϵ. The second function is the "half step" function

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ \sigma_{max} & x > \epsilon \\ \left(\frac{x}{\epsilon} - \frac{1}{\pi}\sin\left(\frac{\pi x}{\epsilon}\right)\right)\sigma_{max} & x \in [0, \epsilon] \end{cases} \tag{5.36}$$

The shape of function (5.35) for different values of ϵ and with $\sigma_{max} = 1$ can be seen in figure 5.11(a). A comparison between the three types of transfer functions mentioned in this chapter is shown in figure 5.11(b). An extensive evaluation of these functions for use with PML boundaries is provided in section 8.1 in Paper III. The result of this study indicates that equation (5.35) with $\epsilon = 1$ provides better results than both equation (5.5) and (5.36). The differences are however fairly small and our experience suggests that any transfer function that is reasonably smooth will work well with our PML boundary.

5.7.5 Estimating Maximum Dampening

Equation (5.11) indicates that the PML boundary condition can theoretically produce an exponentially decaying solution in the boundary region.

Thus it can be assumed that the height of an incident surface wave will be dampened exponentially as it travels through the domain. Hence the theoretical amount of dampening needed to reduce the amplitude of a reflected complex *wave component* by a factor D can be estimated as

$$1 - D = e^{-\frac{k}{\omega}\int_{x_0}^{x_1}\sigma_{max}\sigma(x)dx} \Rightarrow \tag{5.37}$$

$$\sigma_{max} = -\frac{\ln(1-D)}{\frac{k}{\omega}\int_{x_0}^{x_1}\sigma(x)dx}, D \in [0,1[\tag{5.38}$$

where $|x_1 - x_0|$ is the width of the relevant boundary zone and D is the desired dampening given as a fraction. $D = 0.9$ means that the boundary should reduce the incoming wave amplitude by 90% before it hits the outer wall. This scheme requires the calculation of the fraction $\frac{k}{\omega}$, i.e. the (inverse) phase velocity, which can be non-trivial. However, a number of assumptions can be made in order to reduce this problem:

In a non-dispersive medium the phase velocity of all components of a wave packet is the same. Assuming that the fluid is reasonably close to non-dispersive the assumption can thus be made that $\frac{\omega}{k} = C$ for some constant C that can be experimentally estimated or potentially computed. We can also take this simplification even further by assuming $C = 1$ and instead estimating optimal dampening D through experiments. This method is obviously rather approximate and may seem unnecessary since the result is that we now need to experimentally estimate D instead of σ_{max}. However, this method still allows the amount of wave reflection to be specified in terms of amplitude reduction instead of a rather unintuitive number. Making the control of our PML boundaries more intuitive should prove useful when these boundaries are used for visual effects and fluid animation.

Note that equation (5.38) is true for a single wave component while an actual wave packet typically includes a distribution of such components. Thus for equation (5.38) to be accurate it should be related to the fastest moving wave component or alternatively the wave component that carries the largest amplitude.

5.7.6 Solving the PML Equations

Basically, solving the PML equations in the dampening region Ω_d requires solving the equations (5.27a) - (5.27h) where the focus in this thesis is limited to conservative external forces. As can be seen from equation (5.31a) this set of equations is closely related to the regular incompressible Navier-Stokes equations. As a result it is reasonable to assume that a solution algorithm for the incompressible Navier-Stokes equations, such as the Stable Fluids algorithm, can be adapted to solve the boundary version of these equations.

A core component of the Stable Fluids approach is the operator splitting scheme where each term of the Navier-Stokes equations is integrated in sequence, thus allowing specialized solution methods to be applied to each term (see section 2.6.1). Based on this idea we split the PML equations into four solution steps, each of which relates closely to the Stable Fluids algorithm:

Dampened Self-Advection This step relates to the self-advection step (equation (2.42)) in the regular Stable Fluids algorithm. The fluid in the PML boundary region is however dampened as it moves.

Dampened Velocity Diffusion This step relates to the velocity diffusion (equation (2.43)) of the Stable Fluids method. However, the PML boundary equations also contains dampening terms related to the $\mathbf{r}_{\{x,y,z\}}$ auxiliary variables thus affecting the viscous forces of the fluid.

External Forces For conservative external forces the PML version of this term it essentially identical to the standard Stable Fluids approach.

Enforcing Incompressibility Equation (5.31b) indicates that the fluid is potentially divergent in the boundary region. Thus the enforcement of incompressibility will need to be adapted from the traditional pressure projection scheme of the Stable Fluids algorithm (i.e. equations (2.45a) and (2.45b)).

By making use of the operator splitting approach of the Stable Fluids method the scheme for updating the auxiliary tensor field $\mathbf{Q} \equiv \{\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z\}$ one timestep Δt can be summarized as

$$\mathbf{Q}_t \overset{dampened\ advection}{\rightarrow} \mathbf{Q}_1 \overset{viscosity}{\rightarrow} \mathbf{Q}_2 \overset{forces}{\rightarrow} \mathbf{Q}_3 \overset{project}{\rightarrow} \mathbf{Q}_{t+\Delta t} \tag{5.39}$$

where $\mathbf{Q}_t$ is the solution at the beginning of the timestep and $\mathbf{Q}_{t+\Delta t}$ is the new solution to the PML equations at time $t + \Delta t$.

Dampened Self-Advection

In the integration scheme proposed by equation (5.39) the intermediate tensor field $\mathbf{Q}_1$ can be obtained from $\mathbf{Q}_2$ by accounting for dampening and velocity driven advection. Since $\mathbf{v}$ is related to $\mathbf{q}_{\{x,y,z\}}$ through equation (5.27a) this can be seen as self-advection as well. At this point it is useful to treat the velocity related components of the $\mathbf{q}^{\mathbf{v}}_{\{x,y,z\}}$ vectors independently of the density related components $q^{\rho}_{\{x,y,z\}}$, i.e. the first component of each $\mathbf{q}$ vector. This split is equivalent to the common way of writing the Navier-Stokes momentum equations separate from the mass conservation equation (see equations (2.21a) and (2.21b)). The resulting equations become

$$\frac{\partial \mathbf{q}_x^{\mathbf{v}}}{\partial t} + \sigma_x \mathbf{q}_x^{\mathbf{v}} + \frac{\partial v_x \mathbf{v}}{\partial x} = 0 \tag{5.40a}$$

$$\frac{\partial \mathbf{q}_y^{\mathbf{v}}}{\partial t} + \sigma_y \mathbf{q}_y^{\mathbf{v}} + \frac{\partial v_y \mathbf{v}}{\partial y} = 0 \tag{5.40b}$$

$$\frac{\partial \mathbf{q}_z^{\mathbf{v}}}{\partial t} + \sigma_z \mathbf{q}_z^{\mathbf{v}} + \frac{\partial v_z \mathbf{v}}{\partial z} = 0 \tag{5.40c}$$

for the velocity and

$$\frac{\partial q_x^{\rho}}{\partial t} + \sigma_x q_x^{\rho} + \frac{\partial v_x}{\partial x} = 0 \tag{5.41a}$$

$$\frac{\partial q_y^{\rho}}{\partial t} + \sigma_y q_y^{\rho} + \frac{\partial v_y}{\partial y} = 0 \tag{5.41b}$$

$$\frac{\partial q_z^{\rho}}{\partial t} + \sigma_z q_z^{\rho} + \frac{\partial v_z}{\partial z} = 0 \tag{5.41c}$$

for the density components. Currently we have not been able to construct an efficient implicit or semi-Lagrangian[13] scheme for integrating equations (5.40a) - (5.41c). Thus for the scope of this thesis we focus on explicit integration of these equations. Similarly to implicit dampening equations (5.40a) - (5.41c) introduce terms proportional to the potentially large dampening function σ. This can however result in relatively small timesteps in order to achieve stable explicit integration for large σ. In order to obtain a Stable-Fluids based fluid animation system with a stability condition that is not adversely affected by σ (for $\sigma \geq 0$) the following two methods can be applied:

Splitting and Partial Implicit integration A similar stability problem to the one encountered here was encountered for implicit dampening (see section 5.6). By using the operator splitting approach the dampened advection equations above can be split into dampening equations on the form $\frac{\partial q}{\partial t} + \sigma q = 0$ which can be solved through implicit integration (see equation (5.8)) and advection type equations that can be solved through explicit integration. Consequently the resulting scheme is not adversely affected by large σ values.

Modeling the Dampening Behavior The dampening behavior of equation (5.40a) - (5.41c) can be modeled and the result can then be integrated. The resulting scheme will be such that it is not adversely affected by large σ values. This approach is presented in section 6.1 of Paper III.

Both methods presented above do however require the advective component of the PML equations to be solved through explicit integration. As a result

[13]Solution through the Method of Characteristics.

the final integration scheme will not be unconditionally stable. However, in Paper III it was found that in practice a CFL condition is often sufficient for stability when equation (5.40a) - (5.40c) are approximated as pure advection, i.e. the assumption is made that $\frac{\partial v_x \mathbf{v}}{\partial x} = v_x \frac{\partial \mathbf{v}}{\partial x}$ where v_x is assumed to be constant during each integration step. For specific details on the modeling based integration approach for the dampened advection equations see section 6.1 in Paper III.

Velocity Diffusion

Due to the operator splitting of the Stable Fluids method the effect of viscous stress results in a parabolic type equation (see equation (2.43)). Implicit time integration of equation (2.43) results in a Poisson equation which when discretized can be solved efficiently using for example the Conjugate Gradient method. The same method is however not directly applicable to the PML equations due to the additional $\mathbf{r}_{x,y,z}$ vectors now present in the viscous stress tensor. The PML equivalent to equation (2.43) becomes

$$\frac{\partial \mathbf{q}_x}{\partial t} = \nu \frac{\partial \frac{\partial \mathbf{v}}{\partial x} - \sigma_x \mathbf{r}_x}{\partial x} \tag{5.42a}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} = \nu \frac{\partial \frac{\partial \mathbf{v}}{\partial y} - \sigma_y \mathbf{r}_y}{\partial y} \tag{5.42b}$$

$$\frac{\partial \mathbf{q}_z}{\partial t} = \nu \frac{\partial \frac{\partial \mathbf{v}}{\partial z} - \sigma_z \mathbf{r}_z}{\partial z} \tag{5.42c}$$

which by using equation (5.27a) can also be written as

$$\frac{\partial \mathbf{v}}{\partial t} = \nu \nabla^2 \mathbf{v} - \nu \left(\frac{\partial \sigma_x \mathbf{r}_x}{\partial x} + \frac{\partial \sigma_y \mathbf{r}_y}{\partial y} + \frac{\partial \sigma_z \mathbf{r}_z}{\partial z} \right) \tag{5.43}$$

Equation (5.43) makes it clear that the behavior of equations (5.42a) - (5.42c) is essentially the original velocity diffusion of the Stable Fluids method with a number of dampening related correction terms coupled to the auxiliary variables $\mathbf{r}_{\{x,y,z\}}$. In order to solve equations (5.42a) - (5.42c) in a fast and efficient manner we now have two options:

Operator Splitting Use the operator splitting approach to integrate the $\mathbf{r}_{\{x,y,z\}}$ components of equation (5.43) separately, for example using explicit integration. Then solve the remaining poisson equation through implicit integration as is done n the original Stable Fluids algorithm.

Approximate Dampened Diffusion Make the assumption that the contribution of the dampening related correction terms are small enough that visually pleasing results can be obtained also when they are ignored, i.e. assume that $\mathbf{r}_{\{x,y,z\}} = \mathbf{0}$. This essentially assumes low viscosity and is the approach taken in Paper III.

Assuming $\mathbf{r}_{\{x,y,z\}} = \mathbf{0}$ results in one major advantage over the operator splitting approach: The omission of these variables reduces the number of unknowns in the PML equations by 18, greatly decreasing the memory and computational cost of using PML boundaries. Thus this approximate approach is the preferred method in this thesis. Note that our PML boundaries can still be effective also for viscous fluids as demonstrated in section 8.2 of Paper III.

External Forces

Integrating the effect of external forces[14] in the dampening region is straightforward. The force integration equations become

$$\frac{\partial \mathbf{q}_x}{\partial t} = -\mathbf{U}_x \tag{5.44a}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} = -\mathbf{U}_y \tag{5.44b}$$

$$\frac{\partial \mathbf{q}_z}{\partial t} = -\mathbf{U}_z \tag{5.44c}$$

which can for example be solved using simple first order forward Euler integration. For non-conservative external forces the corresponding equations become

$$\frac{\partial \mathbf{q}_x}{\partial t} = \mathbf{a}_x + \sigma_x \frac{\partial \mathbf{a}_x}{\partial t} \tag{5.45a}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} = \mathbf{a}_y + \sigma_y \frac{\partial \mathbf{a}_y}{\partial t} \tag{5.45b}$$

$$\frac{\partial \mathbf{q}_z}{\partial t} = \mathbf{a}_z + \sigma_z \frac{\partial \mathbf{a}_z}{\partial t} \tag{5.45c}$$

which are a bit more complicated. However, since $\mathbf{a}_{\{x,y,z\}}$ can be assumed to be independent of $\mathbf{q}_{\{x,y,z\}}$ simple forward Euler integration can be used here as well. Note that small time-steps or higher order methods may still be necessary to accurately capture the time-dependency of $\mathbf{a}_{\{x,y,z\}}$.

Enforcing Incompressibility

Similarly to the original Stable-Fluids algorithm the successive integration of dampened advection, velocity diffusion and external forces typically result in a velocity field that is not mass conserving. For incompressible flows, mass conservation becomes equivalent to local and global volume conservation which requires that the divergence of the fluid velocity field is locally zero everywhere in the fluid domain. The Stable Fluids algorithm

[14]To be accurate we are integrating external accelerations, however in the incompressible case external forces and external acceleration are trivially related by scaling with the constant fluid density ρ.

solves this problem by projecting the potentially divergent velocity field onto its divergence free part by means of the Helmholtz-Hodge decomposition. As can be seen in equation (5.31b) the PML boundary is however not necessarily divergence free. The PML equations describe a situation where mass can be absorbed and reintroduced through the density related components of the **q** vectors. However, the additional terms in equation (5.31b) has the appearance of source and sink terms. Consequently we treat these terms as sources and sinks during projection, thus projecting **v** onto a potentially divergent solution in the boundary region.

Using first order time discretization this projection can be realized by solving the equation

$$\frac{\Delta t}{\rho}\nabla^2 p = \nabla \cdot \mathbf{v} + \sigma_x \mathbf{B}\mathbf{q}_x + \sigma_y \mathbf{B}\mathbf{q}_y + \sigma_z \mathbf{B}\mathbf{q}_z \tag{5.46}$$

in order to obtain the field p that can be interpreted as pressure. The diagonal, i.e. pressure related components of velocity tensor field can then be modified through

$$\frac{\partial \mathbf{q}_x}{\partial t} = -\mathbf{P}_x \frac{1}{\rho}\frac{\partial p}{\partial x} \tag{5.47}$$

$$\frac{\partial \mathbf{q}_y}{\partial t} = -\mathbf{P}_y \frac{1}{\rho}\frac{\partial p}{\partial y} \tag{5.48}$$

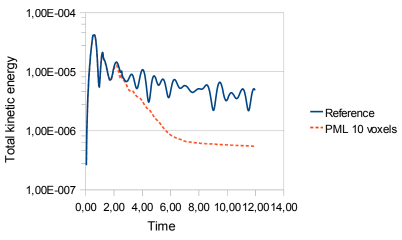
$$\frac{\partial \mathbf{q}_z}{\partial t} = -\mathbf{P}_z \frac{1}{\rho}\frac{\partial p}{\partial z} \tag{5.49}$$

where

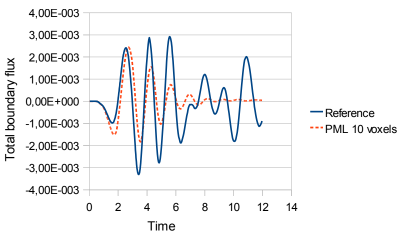
$$\mathbf{P}_x = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \ \mathbf{P}_y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \ \mathbf{P}_z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.50}$$

5.7.7 Evaluation

The PML based non-reflecting boundary condition described in this section is significantly more complex than both the explicit and implicit dampening described in sections 5.5 and 5.6. However, the directional and pseudo-compressible nature of the PML boundary allows for significantly improved performance in preventing boundary reflections of surface waves. In figure 5.12 the kinetic energy and flux graphs are shown for the "pebble in the pond" test simulation (see section 5.4). Notice the almost completely undisturbed decaying oscillation in figure 5.12(b) indicating that virtually no boundary reflections are present that breaks the symmetry of the flow. In figure 5.13 surface offset visualizations are presented comparing the different non-reflecting boundaries presented in this thesis. Here an almost completely undisturbed ring-like wave pattern is exhibited when the PML



(a)



(b)

Figure 5.12: This figure shows a comparison between the undampened "pebble in the pond" reference simulation and the same simulation using PML non-reflecting boundaries with Ω_d 10 grid-points wide and using the smooth step (equation (5.35)) transfer function with $\sigma_{max} = 77$. The dampened advection model described in section 5.7.6 is also used. Figure 5.12(a) shows the kinetic energy of the simulation, 5.12(b) shows the total boundary flux.

boundaries are used. This is very close to the expected behavior if the simulation was indeed part of a much larger fluid domain.

Speedboat

The "speedboat" scenario is a simulation of a fast boat-like object traversing the simulation domain. For this scenario the level set boundaries presented in section 5.3.2 have been used in order to construct a bottle-shaped PML boundary region with two small openings allowing the boat to pass in and out of the domain. An illustration of this boundary is presented in figure 5.14. Figure 5.15 shows frames from a rendering of this animation and figure 5.16 shows the surface offset visualizations comparing the result with an undampened reference simulation. In figure 5.15 the PML boundary region has been removed from the simulation using Constructive Solid Geometry (CSG) operations which can easily be applied to level set geometry (see for example the work by Museth et al. [2002]).

Crazy Boat

The "crazy boat" simulation is intended to show that complex fluid interactions are possible when using PML boundaries. In this scenario a boat moves around in a lake making tight turns in a flower-like pattern. A rendering of this scenario is shown in figure 5.17 and the surface offset visualizations are presented in figure 5.18.

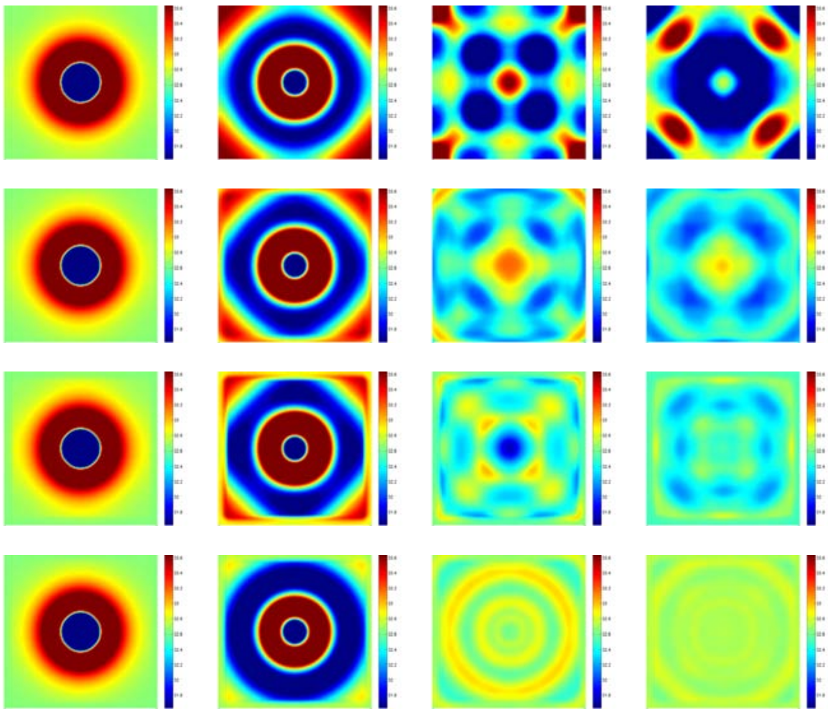


Figure 5.13: This figure shows a comparison between different non-reflecting boundaries presented in this thesis. The depicted simulations are presented in the following order, top to bottom: The undampened reference simulation (top row), explicit dampening, implicit dampening and PML non-reflecting boundaries (bottom row) with Ω_d 10 grid-points wide for all simulations. The visualizations (left to right) correspond to times 1.1, 2.4, 7.2 and 10 seconds into the fluid animation.

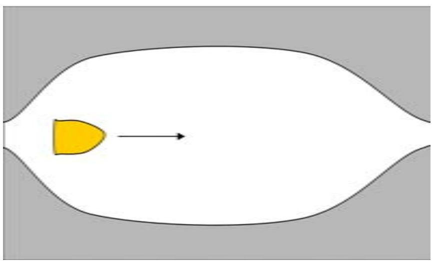


Figure 5.14: Illustration of the non-trivial PML boundary shape used for the “speedboat” scenario.

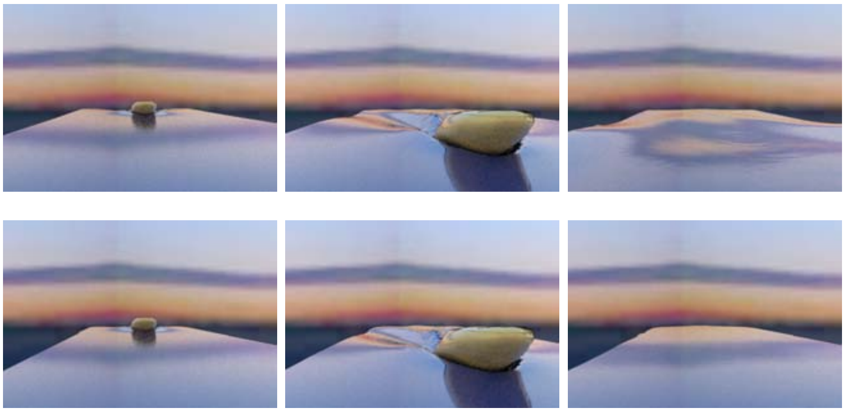


Figure 5.15: Rendering of three frames from the "speedboat" reference simulation. The top row shows the undampened reference simulation and the bottom row shows the same animation using PML boundaries.

Memory Requirements and Computational Cost

Before discussing the memory and computational cost of our PML boundaries we should establish a baseline by estimating the cost for achieving similar results *without* the use of these boundaries. Non-reflecting boundaries using the Navier-Stokes equations without any modifications can be realized as follows: Create a buffer region Ω_d around the simulation of a width such that no wave that crosses $\partial\Omega_f$ has time to traverse Ω_d, reflect off of $\partial\Omega_d$ and return through $\partial\Omega_f$ before the end of the simulation. The necessary width of Ω_d is obviously problem dependent and thus we will use a specific scenario, the "Pebble in the Pond" scenario as an example:

The first wave of the "Pebble in the Pond" simulation reaches the edge of the simulation domain after approximately 2 seconds. Thus the full 12 second animation would require a domain that is 3.5 times[15] as wide. This in turn increases the fluid volume in this scenario by 12.25 times resulting in a minimum[16] 1125% increase in memory footprint and simulation time.

The explicit and implicit boundaries are more efficient than the above "brute force" approach, however the results in sections 8.1.5 and 8.1.6 of Paper III indicate that they require both more memory and computation time than our PML boundaries given that high quality[17] results are desired.

[15]The ring-like wave covers half the width of the domain in all directions in approximately 2 seconds. Thus to avoid visible reflections it must reach $\partial\Omega_d$ after more than (12-2)/2 = 5 seconds.

[16]Assuming that all computations scale linearly with volume. This is however typically not true since the Conjugate Gradient method requires more iterations to converge to a desired precision as the number of unknowns increase. As a result the computational cost increases faster than linear with increased fluid volume.

[17]I.e. low amounts of boundary reflection.

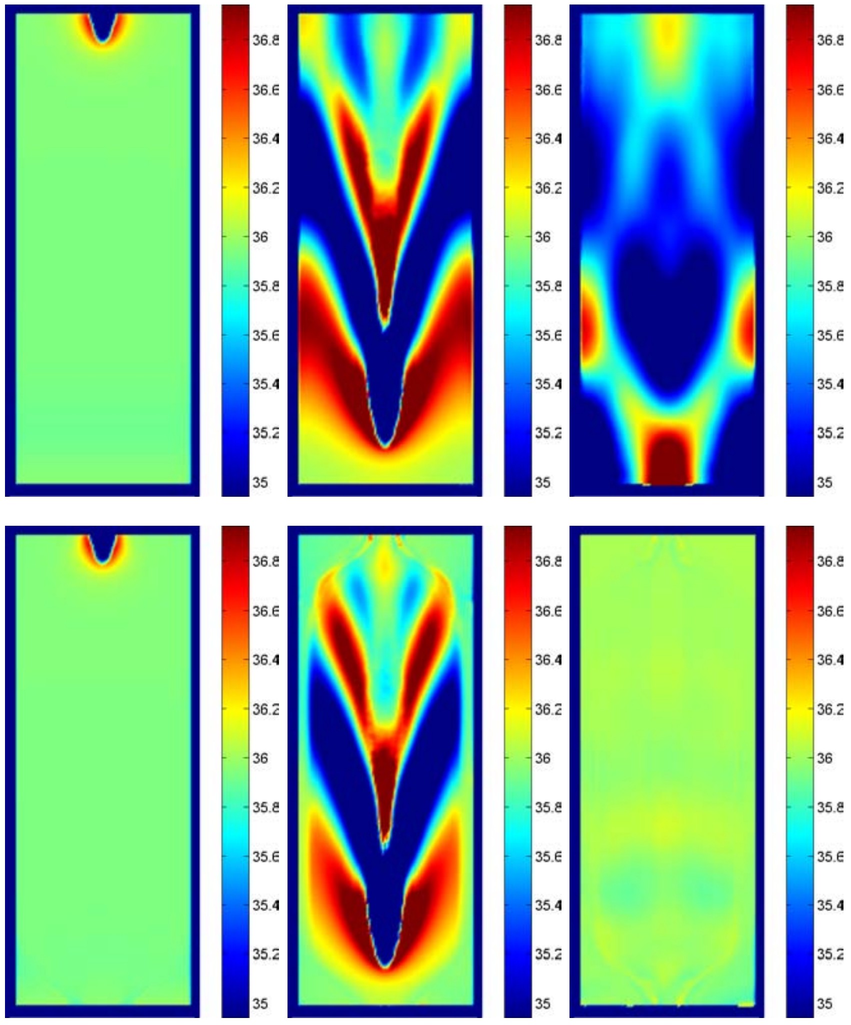


Figure 5.16: Surface offset visualization of the "Speedboat" simulation. The first three images (top row) show the reference simulation after 0.8, 2.5 and 7.8 seconds respectively. The three images on the second row show the same simulation with a PML wave absorbing layer present.

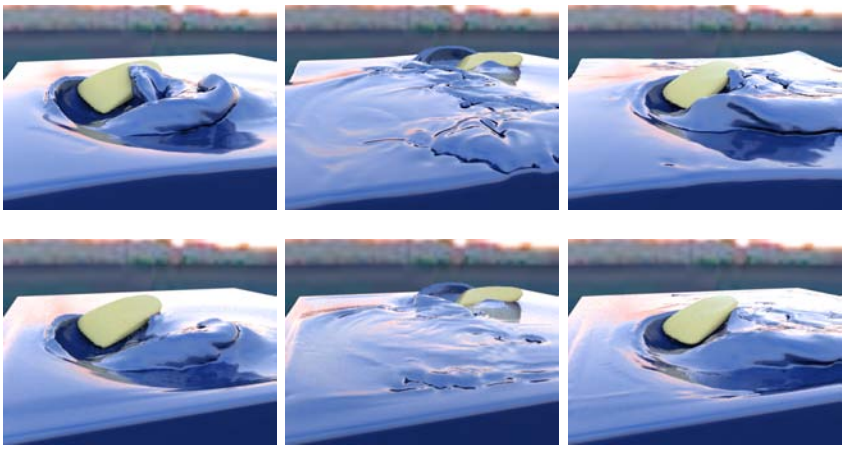


Figure 5.17: Rendering of three frames from the "crazy boat" reference simulation. The top row shows the undampened reference simulation and the bottom row shows the same animation using PML boundaries.

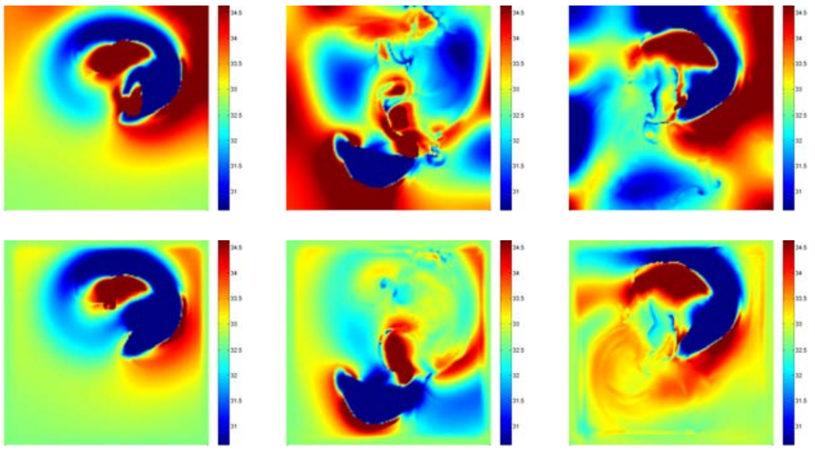


Figure 5.18: Surface offset visualization of the "crazy boat" simulation. The first three images (top row) shows the reference simulation after 1.4, 5.8 and 11.8 seconds respectively. The three images on the second row shows the same simulation with a PML wave absorbing layer present.

The memory requirement of the full set of PML equations as derived in section 5.7.1 is fairly high when compared to the regular incompressible Navier-Stokes equations. The number of equations and unknowns increase from four to 30 indicating a 650% increase in memory required to store the related variables. This number is even higher if non-conservative external forces are present. However, if the viscosity-related dampening is omitted as is done in our Stable-Fluids based solution algorithm the number of equations and unknowns decrease by 18, leaving a 200% estimated increase in memory usage. However, the full PML equations are only needed in the actual boundary regions which due to the effectiveness of these boundaries are typically small. For the "pebble in the pond" experiment presented in section 5.7.7 a boundary width of 6% of the width of the simulation domain was sufficient[18]. This boundary correspond to 23% of the total fluid volume and thus the PML boundary will results in an approximatly 45% increase in memory usage[19] for this scenario when compared to the same domain without any dampening.

Our solution algorithm for the PML equations is very similar to the Stable-Fluids approach. Thus the computational cost of our method per iteration of the solution algorithm scales approximately with the number of additional variables when compared to the undampened Stable-Fluids method. The theoretical increase in computation time compared to the undampened Navier-Stokes equations is thus 200%, although if the PML equations are only solved in the boundary region the performance penalty will be lower than this. A comparison of the relative performance of the explicit, implicit and PML boundaries can be found in table I in Paper III.

5.8 PML Boundaries on MAC Grids

The work presented in Paper III is primarily focused on solving the PML boundary equations on regular computational grids with velocity vectors stored at the center of each grid-cell. However, PML boundaries can readily be applied to MAC grids using the solution algorithm presented in section 5.7.6 as well as section 6 of Paper III. We suggest the following method for splitting the PML $\mathbf{q}_{\{x,y,z\}}$ vectors onto a MAC grid:

The first row of $\mathbf{q}_x$, $\mathbf{q}_y$ and $\mathbf{q}_z$ correspond to density components. These values are thus reasonable to store at cell-centers as a three-element vector. The second row of $\mathbf{q}_x$, $\mathbf{q}_y$ and $\mathbf{q}_z$ correspond to velocity components in the Cartesian x-direction. These values will thus be stored together as a three-element vector on x faces of each grid-cell. Equivalently the third and fourth row of the $\mathbf{q}_{\{x,y,z\}}$ vectors can be collected into three-element vectors stored on the y and z faces respectively.

[18]Note that in general the optimal boundary width for a given scenario is problem dependent and is typically coupled to both incident wave amplitude and velocity.

[19]Provided that the auxilliary variables are only stored in Ω_d.

Using this construction finite difference computations can be performed on the staggered velocity tensor in an equivalent manner to what is used for the velocity components of a regular velocity MAC grid. Note that the three components stored on the faces of each grid-cell can be added together through equation (5.27a) to recreate the x, y and z components of the regular velocity vector.

5.9 Summary

The main goal of the work presented in Paper III and this chapter was to investigate the possibility of creating non-reflective boundaries in order to make a local free surface fluid simulation behave as if surrounded by a larger open fluid domain. This approach will then allow potentially large amounts of memory and computation time to be saved by reducing the simulation domain required to compute accurate fluid animations in a local region of a larger fluid. To this end we have constructed and presented three types of non-reflective boundaries. The first two, explicit and implicit dampening, are simple to implement and the resulting boundary equations can be solved in an unconditionally stable manner using a slightly modified Stable-Fluids algorithm. However these methods typically require large dampening regions to be added to the simulation domain in order to remove most of the amplitude from reflected waves. This makes explicit and implicit dampening fairly slow and memory demanding in practice.

In order to provide higher efficiency as well as more physically motivated dampening we have developed a PML based wave *absorbing* boundary condition. Our PML boundary can drastically reduce boundary wave reflections also with fairly small boundary regions. The method is however significantly more complex than explicit and implicit dampening.

In order to solve the PML equations in a fast and efficient manner we have presented a solution method based on the Stable-Fluids algorithm. Our PML solver is however only conditionally stable although the stability condition can be made independent of the parameters of the boundaries.

Chapter 6

Conclusions

The overall aim of this thesis was to improve and extend the available toolset for Eulerian free surface fluid animation and level set based surface tracking with a particular focus on memory efficient methods. For this purpose the memory efficient fluid animation system presented in section 2.6 was first constructed. This system was then further refined by reducing the memory requirements of the level set and PLS surface tracking methods through compression as described in Paper I. A further reduction of memory usage was enabled through the out-of-core streaming approach also presented in Paper I.

At this point it was clear that PLS surface tracking, even when using compression, is a very memory demanding method. In order to provide enhanced level set surface tracking without resorting to the PLS method the dual-resolution approach of Paper II was developed. The resulting scheme provides enhanced surface tracking compared to the level set method using up to 94% less memory than the PLS method.

At this point focus was put on reducing the memory requirements of detailed ocean simulations by reducing the actual simulation space. An investigation was started into appropriate boundary conditions for allowing the local simulation of a larger open fluid domain. This project resulted in three different non-reflecting boundary conditions for use with incompressible free surface fluid animation. The most effective of these boundary conditions was also the focus of Paper III.

Below follows a more in-depth discussion on the main findings, future work and potential impact of each of the primary projects of this thesis.

6.1 The Out-Of-Core and Compression Framework (Paper I)

6.1.1 Main Findings

The main goal of this project was to attempt to reduce the memory footprint of free surface fluid animations using the PLS method by means of compression and out-of-core streaming. Focus was primarily put on the memory demanding PLS method since it was found to use close to an order of magnitude more data than the rest of our already compact fluid animation system.

Compressed Fluid Animations

Using compression for fluid animation was found to be a computationally expensive process best suited for situations where primary memory is a very limiting factor. The exception is however our compression of the PLS particles. Using a quantization based approach we were able to reduce the memory requirement for these particles by 70% at a 5% performance penelty.

Out-of-Core Fluid Animations

Out-of-core streaming is a viable approach for free surface fluid animation on DT and DV grids. Our results indicate that both level set and PLS computations can be performed with less than a factor two increase in simulation time when streaming data from a standard consumer grade hard disk drive. This suggests that close to in-core performance should be possible when using modern SSDs in RAID configuration. A notable exception is however the data intensive Conjugate Gradient method that is not well suited for out-of-core computing. Out-of-core streaming of PLS particles can also be combined with our fast particle compression method, effectively resulting in a more than 200% increase in data throughput.

6.1.2 Future Work

When focusing on low memory footprints for this project we have consequently put less emphasis on another important factor in fluid animation: computational speed. Although our framework allows essentially arbitrary resolution for level set geometry computations on high resolution geometry is time-consuming. Consequently parallel computing on compressed and out-of-core level set geometry is an interesting possibility. For this purpose SSDs may prove highly useful.

Also we have chosen not to compress the fluid velocity fields in our system for performance reasons. However, recent work on fast floating-point compression methods, such as the method presented by Burtscher and Ratanaworabhan [2007], seems highly appropriate for use with our framework. Such methods can be used to compress fluid velocity and scalar fields and also potentially replace our compression method for the level set distance field. Furthermore, if fast enough, this type of compression can potentially increase the effective data throughput for out-of-core fluid computations, thus increasing overall performance.

Although the work presented in this thesis focuses on uniformly sampled Eulerian grids the possibility of combining our out-of-core and compression schemes with non-uniform sampling is also interesting. The resulting system would allow highly compact data to be adaptively distributed where it is needed the most, thus increasing memory efficiency even further.

6.1.3 Potential Impact

We see a number of areas in which we expect our out-of-core and compression framework to prove useful:

High Resolution PLS Surface Tracking The fast compression and out-of-core streaming of PLS data allows for a significant reduction in the memory footprint for fluid animations inducing a small penalty to performance. Futhermore the particle compression scheme is fairly simple to implement on DT-Grids and we expect that this is also the case on other sparse grids like octrees and DB grids. Consequently we expect our compression method to be a useful and widely adopted addition to the PLS method.

Extreme Resolution Level Sets Using our out-of-core and compression framework level set geometry of essentially aribtrary resolution can be represented. This allows a level of detail often only associated with explicit meshes. Geometric deformations, such as advection and CSG operations, can still be performed on this high resolution geometry at a small to moderate performance penalty. Consequently we expect our framework to be highly useful for high-detail modeling of large and/or complex scenes of level set geometry, scenes that can then be used in for example fluid animations.

Fast Out-of-Core Computing on DT-Grids The arrival of fast and relatively cheap secondary storage devices like Solid State Drives (SSDs) currently makes cheap high-performance out-of-core computing on DT-Grids a very real possibility. Judging by the performance we were able to obtain using a single hard disk drive we expect that a RAID of SSDs should allow our framework to maintain in-core performance for many scenarios. Consequently our out-of-core and compression framework is probably more attractive now than ever before.

6.2 The Dual Resolution Surface Tracking Method (Paper II)

6.2.1 Main Findings

The main goal with this project was to develop a memory efficient Eulerian alternative to the PLS method using level sets at two levels of resolution. Similarly to previous work in this field we discovered that a level set based dual resolution interface tracking method will have problems with aliasing artifacts caused by downsampling the high resolution geometry. Our approach to resolving this issue was to design two geometrically adaptive filters (the SAM and SP-SAM filters) which are applied before the downsampling is performed. We found that our filters allow plausible animation to be computed for the high resolution geometry while behaving in a more physically accurate manner than the simpler Liquid Biased filter [Kim et al., 2009], avoiding boundary penetration and sampling related topology changes.

By using high order accurate advection schemes we were also able to reach surface tracking results subjectively similar to the PLS method using up to 94% less memory. Arguably our results were even better than the PLS is several scenarios. However, our Eulerian approach is still unable to match the high advection accuracy of particles. Our method was also found to be more computationally expensive than the PLS approach. In our tests a 14%-120% performance penalty was measured depending on the ratio between the fluid volume and the area of the fluid surface.

6.2.2 Future Work

This project has in our experience worked quite well as an alternative to, and often even a replacement for, the PLS method and we have used it extensively. However, we still see several areas where our method can be developed further:

Due to the downsampling process the fluid solver is only able to "see" a rather approximate version of the high resolution geometry. However, by accurately estimating the actual fraction of a low resolution grid-cell that is occupied by the high resolution surface we expect that a more accurate coupling can be achieved. For this purpose a fluid animation system based on volume fractions [Batty et al., 2007; Bridson, 2008] may prove useful.

Similarly the trilinear interpolation used when upsampling the solution to the Navier-Stokes equations may introduce advection artefacts. Thus it is possible that more advanced, possibly physics based[1], interpolation can provide more accurate animation of the high resolution surface. Here volume fractions can potentially also prove useful.

Finally the linear interpolation along a stencil used to estimate surface thickness in our method can be made more accurate. For example a ray can be traced along the surface normal of the geometry and ray/level set intersections can be computed based on trilinear (instead of linear) interpolation in each voxel[2].

6.2.3 Potential Impact

The SAM Filter The adaptive SAM filter is a logical next step from the robust but simple LB filter. The SAM filter is also fairly simple to

[1]For example by modeling the behavior of the fluid on a subvoxel level.

[2]In this context a voxel is a grid cell with grid points in each corner.

implement and thus we expect it to replace the LB filter for use with dual resolution implicit surface tracking.

The SP-SAM Filter The SP-SAM filter is essentially able to provide user-control over the presence of fluid sheets in fluid animations and as such should be a highly useful tool for many visual effects oriented scenarios. For example we expect the SP-SAM filter to be useful for Eulerian animation of splashes, waterfalls and similar phenomena, especially if combined with particle based methods for the animation of droplets.

Our Dual-Resolution Surface Tracking Method We consider our method to be an effective alternative to the PLS method, however we also suspect that it will provide a convenient, robust and reliable replacement for PLSs in many Eulerian fluid animation scenarios.

Fluid Animation Previews The SAM filter allows the Navier-Stokes equations to be solved at a lower resolution than what is used for the surface tracking algorithm. Thus we see the possibility of our method being useful for computing fast previews of high resolution free surface fluid animations.

6.3 The Non-Reflecting Boundaries (Paper III)

6.3.1 Main Findings

The primary aim of the non-reflecting boundary project was to implicitly reduce both the memory footprint and the computational cost by allowing for accurate local simulations of a larger free surface fluid.

We found that this approach was indeed viable by deploying non-reflecting boundaries in buffer regions surrounding the local fluid domain. Although this was found to be possible using relatively simple boundary equations the simple methods required large boundary regions to be effective. As a result it was found that simple non-reflecting boundaries in practice could be both computationally expensive and memory demanding.

For this reason we investigated the possibility to use more advanced boundaries based on the PML approach. These boundaries were found to be significantly better at preventing wave reflections than simpler methods when using narrow boundary regions. We were also able to solve the relatively complicated PML boundary equations in a conditionally stable manner using a modified Stable Fluids solver. Although this solution algorithm can be improved our approach demonstrates that this type of boundary is viable for use with free surface fluid animations.

6.3.2 Future Work

We see potential for further development of the explicit dampening method. In its current form this approach dampens all wave motion by the time the wave has reached the outer boundary. However, by calculating an appropriate target velocity function at the outer boundary it should theoretically be possible to ensure that most of the wave amplitude has been remove when the wave returns to the inner boundary instead of when it hits the outer boundary. Thus the wave can effectively be dampened over a distance of two boundary widths instead of one, potentially doubling the efficiency of explicit boundaries.

Regarding the solution algorithm for the PML equations we have made approximations both concerning the dampened viscous forces and the (non-conservative) external forces in the boundary domain. Both of these areas warrant further investigation. We expect that accurate wave absorption of viscous flow will be slower and less memory efficient due to the additional 18 equations introduced. However, the resulting method should also prove significantly more effective for high viscosity flow. Regarding external forces the derivation in this thesis provides a theoretical foundation for non-conservative forces in the boundary domain. However, these force equations have not yet been extensively tested and this remains an area for future study. Finally our current solution algorithm has a tendency to produce a small residual flux into or out of the boundary region suggesting that the integration of the density components of the boundary can be improved. Here the known quantity that the sum of the density components is the constant density should be useful in detecting and potentially correcting errors.

Both the implicit and PML boundaries require the estimation of the maximum dampenig σ_{max}. This estimation can be improved by estimating the phase velocity and amplitude of the components of an incident wave packet. This can potentially be achieved through a Fourier transform computed on a sliding window of appropriate size following the inner boundary. Another alternative is to project the incident wave onto a subspace of precomputed wave packets of known amplitude and frequency distributions. If this type of information can be obtained it can then potentially be used for adaptive PML boundaries where the width of the boundary domain is adjusted based in the wave components that pass through the inner boundary. A further development of this idea is to also make use of non-uniform sampling. Assuming that the largest amplitudes of the incident wave packets are found at the lower frequencies larger sampling distances can be used further into the boundary. This since high frequency wave components will have been effectively extinguished at this point. Using the ideas presented in section 2.5.6 this approach may be possible while still using finite difference schemes.

Our non-reflecting boundaries can potentially be applied to both the

2D and 3D components of a hybrid ocean simulation system. This can potentially allow great control of the behavior of the boundary regions overlapping the 2D and 3D simulation, in turn creating more realistic and more versatile 2D/3D fluid animation systems.

Our non-reflecting boundaries essentially ”hides“ the physical wall at the outer boundary from the fluid in the interior domain. Consequently the fluid behaves as if the outer boundary is far away. This property should be useful for modeling the behavior of deep water while animating fluids in a fairly shallow domain. By placing a non-reflecting boundary along the bottom of the simulation domain we expect that it is possible to make the fluid behave as if the floor of the simulation is far away.

In computational aeroacoustics forced convection can be used in non-reflecting boundaries in order to counteract waves moving towards the inner boundary (see for example [Richards et al., 2004]). This approach may be interesting to apply to the boundaries presented in this thesis as well.

Finally, the current formulation of our PML equations is written in the split variable formulation. We expect however that they can be rewritten in an unsplit formulation as well, thus avoiding the potential issues with splitting the external force vector into an external force tensor. Furthermore the unsplit formulation could reduce the need for auxiliary variables in the boundary region thus providing higher memory efficiency. However, the resulting equations are expected to be complex and may thus be difficult to solve.

6.3.3 Potential Impact

We expect that our non-reflective boundaries will have a significant impact on the field of incompressible free surface fluid animation. Several researchers and visual effects studios has at the point of writing this thesis expressed an interest in our results and our methods. Below follows a number of specific areas where we expect our boundaries to make an impact:

Awareness of Non-Reflective Boundaries We expect our work to raise the awareness in the computer graphics community of the existence of non-reflective boundaries and the PML method. As such we expect our work to lead the way on future developments concerning such boundaries for use with free surface fluid animation.

Improved Local Ocean Animation Using our boundaries accurate local simulations of ocean surfaces can be performed in a physically plausible manner. Our method allows complex interactions with solid objects on a local patch of ocean without the need to worry about unphysical wave reflection from the edges of the simulation domain.

Improved 2D/3D Fluid Coupling We expect that our method will allow greater control of boundary regions when coupling 2D and 3D fluids. Non-reflective boundaries of the type presented in this chapter, including PMLs, are relatively easy to implement[3] for simplified fluid models, making it possible to use this type of boundaries both for the 2D and 3D parts of a hybrid ocean simulation systems.

6.4 Concluding Remarks

Looking towards the future it is hard to say what impact this thesis and its methods will have. All of our presented methods have proven very useful in our own work. Both our out-of-core and compression framework as well as our dual-resolution surface tracking method has also been received with interest by the computer graphics community. Furthermore several researchers and visual effects studios has at the point of writing this thesis expressed interest in our non-reflective boundaries. However, the field of computer graphics is still in rapid development and methods that are considered useful today may well be surpassed and forgotten when faced with the methods of tomorrow. Thus in the end all that can be said is this:

Each method presented in this thesis is a specific tool for a specific task. As with all tools there are tasks for which these tools are appropriate and tasks for which they are not. However, in scenarios where computational resources are abundant while fast memory is limited the methods presented in this thesis should present an attractive alternative now and well into the future.

[3]See [Richards et al., 2004] for an example of a PML for the linearized Euler equation and [Johnson, 2007] for the wave equation. We have also as part of our research experimented with PMLs for the shallow water equations.

Bibliography

S. Abarbanel and D. Gottlieb. A mathematical analysis of the pml method. *J. Comput. Phys.*, 134(2):357–363, 1997. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1997.5717.

D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118(2):269–277, 1995. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1995.1098.

J. D. Anderson. *Computational Fluid Dynamics: The Basics With Applications*. McGraw-Hill Science Engineering, USA, 1995. ISBN 0-07-001685-2.

A. W. Bargteil, T. G. Goktekin, J. F. O'brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1):19–38, 2006. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1122501.1122503.

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

C. Batty, S. Xenos, and B. Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Proceedings of Eurographics*, 2010.

E. Bcache, S. Fauqueux, and P. Joly. Stability of perfectly matched layers, group velocities and anisotropic waves. *Journal of Computational Physics*, 188(2):399 – 433, 2003. ISSN 0021-9991. doi: DOI:10.1016/S0021-9991(03)00184-0. URL `http://www.sciencedirect.com/science/article/B6WHY-48H8777-1/2/683b66a009b67f7d1e108d7f69f8144d`.

E. Bécache, A.-S. B.-B. Dhia, and G. Legendre. Perfectly matched layers for the convected helmholtz equation. *SIAM J. Numer. Anal.*, 42(1):409–433, 2004. ISSN 0036-1429. doi: http://dx.doi.org/10.1137/S0036142903420984.

J.-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185 – 200, 1994. ISSN 0021-9991. doi: DOI:10.1006/jcph.1994.1159. URL `http://www.sciencedirect.com/science/article/B6WHY-45P0TJR-1P/2/9aa5d47e587908f2e75691914c79ac80`.

J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25 – 38, 1988. ISSN 0010-4655. doi: DOI:10.1016/0010-4655(88)90020-3. URL `http://www.sciencedirect.com/science/article/B6TJ5-46FXB87-4B/2/bb909b136946bfc5f47a61c01547a726`.

R. Bridson. *Fluid Simulation for Computer Graphics.* A K Peters, 2008. ISBN 978-1-56881-326-4.

T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4):1–9, 2010. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1778765.1778784.

M. Burtscher and P. Ratanaworabhan. High throughput compression of double-precision floating-point data. In *DCC '07: Proceedings of the 2007 Data Compression Conference*, pages 293–302, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2791-4. doi: http://dx.doi.org/10.1109/DCC.2007.44.

S. Chen, Z. Wang, X. Shan, and G. D. Doolen. Lattice boltzmann computational fluid dynamics in three dimensions. *Journal of Statistical Physics*, 68:379–400, 1992. ISSN 0022-4715. URL `http://dx.doi.org/10.1007/BF01341754`. 10.1007/BF01341754.

N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-1-59593-624-4.

W. C. Chew and W. H. Weedon. A 3-d perfectly matched medium from modified maxwell's equations with stretched coordinates. *Microwave Opt. Tech. Lett*, 7:599–604, 1994.

Clay Mathematics Institute. Navier-stokes equations, Aug. 2010. URL `http://www.claymath.org/millennium/Navier-Stokes_Equations/`.

T. F. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.*, 190(1):311–324, 2003. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/S0021-9991(03)00276-6.

M. Ellero, M. Serrano, and P. Español. Incompressible smoothed particle hydrodynamics. *J. Comput. Phys.*, 226(2):1731–1752, 2007. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2007.06.019.

D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1): 83–116, 2002a.

D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph.*, 21(3):736–744, 2002b. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/566654.566645.

D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM200345144. ASME*, pages 2003–45144, 2003.

D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures*, 83(6-7):479 – 490, 2005. ISSN 0045-7949. doi: DOI:10.1016/j.compstruc.2004.04.024. Frontier of Multi-Phase Flow Analysis and Fluid-Structure.

Z. Fan, Y. Zhao, A. Kaufman, and Y. He. Adapted unstructured lbm for flow simulation on curved surfaces. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 245–254, New York, NY, USA, 2005. ACM. ISBN 1-7695-2270-X. doi: http://doi.acm.org/10.1145/1073368.1073404.

R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: http://doi.acm.org/10.1145/383259.383260.

C. L. Fefferman. Existence and smoothness of the navier-stokes equations, Aug. 2010. URL `http://www.claymath.org/millennium/Navier-Stokes_Equations/navierstokes.pdf`.

N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th conference on Computer graphics and interactive techniques*, pages 23–30, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: http://doi.acm.org/10.1145/383259.383261.

N. Foster and D. Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5):471–483, 1996. ISSN 1077-3169. doi: http://dx.doi.org/10.1006/gmip.1996.0039.

N. Foster and D. Metaxas. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International*, page 178, Washington, DC, USA, 1997a. IEEE Computer Society. ISBN 0-8186-7825-9.

N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1997b. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: http://doi.acm.org/10.1145/258734.258838.

F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980. doi: 10. 1137/0717021. URL `http://link.aip.org/link/?SNA/17/238/1`.

T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien. A method for animating viscoelastic fluids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 463–468, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1186562.1015746.

T. Hagstrom and I. Nazarov. Absorbing layers and radiation boundary conditions for jet flow simulations. In *8th AIAA/CEAS Aeroacoustics Conference and Exhibit*, pages AIAA paper 2002–2606, 2002.

T. Hagstrom and I. Nazarov. Perfectly matched layers and radiation boundary conditions for shear flow calculations. In *9th AIAA/CEAS Aeroacoustics Conference and Exhibit*, pages AIAA paper 2003–3298, 2003.

T. Hagstrom, J. Goodrich, I. Nazarov, and C. Dodson. High-order methods and boundary conditions for simulating subsonic flows. In *11th AIAA/CEAS Aeroacoustics Conference*, pages AIAA–2005–2869, 2005.

F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965. ISSN 0031-9171.

C. W. Hirt, J. L. Cook, and T. D. Butler. A lagrangian method for calculating the dynamics of an incompressible fluid with free surface. *Journal of Computational Physics*, 5(1):103 – 124, 1970. ISSN 0021-9991. doi: DOI:10.1016/0021-9991(70)90055-0.

B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Trans. Graph.*, 25(1):151–175, 2006. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1122501.1122508.

F. Hu. On the construction of pml absorbing boundary condition for the non-linear euler equations. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, pages AIAA–2006–798, 2006.

F. Q. Hu. On absorbing boundary conditions for linearized euler equations by a perfectly matched layer. *J. Comput. Phys.*, 129(1):201–219, 1996. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1996.0244.

F. Q. Hu. A stable, perfectly matched layer for linearized euler equations in unsplit physical variables. *Journal of Computational Physics*, 173(2): 455 – 480, 2001. ISSN 0021-9991. doi: DOI:10.1006/jcph.2001.6887.

F. Q. Hu. A perfectly matched layer absorbing boundary condition for linearized euler equations with a non-uniform mean flow. *Journal of Computational Physics*, 208(2):469 – 492, 2005. ISSN 0021-9991. doi: DOI:10.1016/j.jcp.2005.02.028.

F. Q. Hu. Development of pml absorbing boundary conditions for computational aeroacoustics: A progress review. *Computers and Fluids*, 37(4):336 – 348, 2008. ISSN 0045-7930. doi: DOI:10.1016/j.compfluid.2007.02.012. Turbulent Flow and Noise Generation.

F. Q. Hu, X. D. Li, and D. K. Lin. Absorbing boundary conditions for nonlinear euler and navier-stokes equations based on the perfectly matched layer technique. *J. Comput. Phys.*, 227(9):4398–4424, 2008. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2008.01.010.

L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Comput. Graph. Forum*, 22(3):343–348, 2003.

G. Johansson, O. Nilsson, A. Söderström, and K. Museth. Distributed ray tracing in an open source environment (work in progress). pages 7–11. Linköping University Electronic Press, 2006.

S. G. Johnson. Notes on perfectly matched layers (pmls), 2007. URL `http://math.mit.edu/~stevenj/18.369/pml.pdf`. online MIT course notes (Aug. 2007).

D. Kim, O.-y. Song, and H.-S. Ko. Stretching and wiggling liquids. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–7, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-858-2. doi: http://doi.acm.org/10.1145/1661412.1618466.

D. Kincaid and W. Cheney. *Numerical analysis: mathematics of scientific computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 1991. ISBN 0-534-13014-3.

A. Klarbring. *Models of Mechanics*. Springer, Dordrecht, The Netherlands, 2006. ISBN 1-4020-4834-3.

B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, 2006. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1141911.1141961.

X. Liu, S. Osher, and T. Chan. Weighted essentially nonoscillatory schemes. *J. Comput. Phys.*, 115:200–212, 1994.

W. Lorenson and H. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proc. SIGGRAPH)*, 21(4): 163–169, 1982.

F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 457–462, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/1186562.1015745.

F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/TVCG.2008.37.

M. Mandal and A. Asif. *Continuous and Discrete Time Signals and Systems*. Cambridge University Press, 2007. ISBN 9780521854559.

V. Mihalef, D. N. Metaxas, and M. Sussman. Textured liquids based on the marker level set. *Comput. Graph. Forum*, 26(3):457–466, 2007.

J. Molemaker, J. M. Cohen, S. Patel, and J. Noh. Low viscosity flow simulations for animation. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 9–18, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. ISBN 978-3-905674-10-1.

J. Monaghan. An introduction to sph. *Computer Physics Communications*, 48(1):89–96, January 1988. ISSN 00104655. doi: 10.1016/0010-4655(88)90026-4. URL `http://dx.doi.org/10.1016/0010-4655(88)90026-4`.

P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun. Energy-preserving integrators for fluid animation. *ACM Trans. Graph.*, 28(3): 1–8, 2009. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1531326.1531344.

K. Museth. An efficient level set toolkit for visual effects. In *SIGGRAPH '09: SIGGRAPH 2009: Talks*, pages 1–1, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-834-6. doi: http://doi.acm.org/10.1145/1597990.1597995.

K. Museth and M. Clive. Cracktastic: fast 3d fragmentation in "the mummy: Tomb of the dragon emperor". In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks*, pages 1–1, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-343-3. doi: http://doi.acm.org/10.1145/1401032.1401110.

K. Museth and C. Leforestier. On the direct complex scaling of matrix elements expressed in a discrete variable representation: Application to molecular resonances. *J. Chem. Phys.*, 104(18):7008–7014, 1996.

K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 21(3): 330–338, July 2002.

K. Museth, M. Clive, and N. B. Zafar. Blobtacular: surfacing particle system in "pirates of the caribbean 3". In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 20, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1278780.1278804.

S. Navti, K. Ravindran, C. Taylor, and R. Lewis. Finite element modelling of surface tension effects using a lagrangian-eulerian kinematic description. *Computer Methods in Applied Mechanics and Engineering*, 147(1-2):41 – 60, 1997. ISSN 0045-7825. doi: DOI:10.1016/S0045-7825(97)00017-0.

D. Neuhasuer and M. Baer. The time?dependent schrdinger equation: Application of absorbing boundary conditions. *J. Chem. Phys.*, 90(8): 4351–4355, 1989.

D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566643.

M. B. Nielsen and K. Museth. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *J. Sci. Comput.*, 26(3):261–299, 2006. ISSN 0885-7474. doi: http://dx.doi.org/10.1007/s10915-005-9062-8.

M. B. Nielsen, O. Nilsson, A. Söderström, and K. Museth. Virtually infinite resolution deformable surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 66, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6.

M. B. Nielsen, O. Nilsson, A. Söderström, and K. Museth. Out-of-core and Compressed Level Set Methods. *ACM Trans. Graph.*, 26(4), 2007. ISSN 0730-0301.

O. Nilsson. *Level-set methods and geodesic distance functions.* PhD thesis, Linkping UniversityLinkping University, Department of Science and Technology, The Institute of Technology, 2009.

O. Nilsson and A. Söderström. Euclidian distance transform algorithms : A comparative study. Technical report, Linköping UniversityLinköping University, Department of Science and Technology, 2007.

NYC Department of City Planning. New york city land use, Aug. 2010. URL `http://www.nyc.gov/html/dcp/html/landusefacts/landusefactshome.shtml`.

S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(88)90002-2.

S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 1 edition, October 2002. ISBN 0387954821. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387954821`.

D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. Kang. A pde-based fast local level set method. *Journal of Computational Physics*, 155(2): 410–438, 1999.

W. P. Reinhardt. Complex coordinates in the theory of atomic and molecular structure and dynamics. *Ann. Rev. Phys. Chem.*, 33:223–255, 1982.

S. K. Richards, X. Zhang, X. X. Chen, and P. A. Nelson. The evaluation of non-reflecting boundary conditions for duct acoustic computation. *Journal of Sound and Vibration*, 270(3):539 – 557, 2004. ISSN 0022-460X. doi: DOI:10.1016/j.jsv.2003.09.042. 2002 I.M.A. Conference on Computational Aeroacoustics.

D. Salomon. *Data Compression: The Complete Reference*. pub-SV, pub-SV:adr, 2007. ISBN 1-84628-602-6. With contributions by Giovanni Motta and David Bryant.

A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073282.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

C. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics, 2002.

A. Söderström and K. Museth. Non-reflective boundary conditions for incompressible free surface fluids. In *SIGGRAPH '09: SIGGRAPH 2009: Talks*, pages 1–1, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-834-6.

A. Söderström and K. Museth. A Spatially Adaptive Morphological Filter for Dual-Resolution Interface Tracking of Fluids. pages 5–8, Norrköping, Sweden, 2010. Eurographics Association. ISBN undefined. URL `http://www.eg.org/EG/DL/conf/EG2010/short/005-008.pdf`.

A. Söderström, M. Karlsson, and K. Museth. A PML Based Non-Reflective Boundary for Free Surface Fluid Animation. *ACM Trans. Graph.*, 29 (5), 2010. doi: 10.1145/1857907.1857912.

J. Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: http://doi.acm.org/10.1145/311535.311548.

J. Steinhoff and D. Underhill. Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, 6(8):2738–2744, 1994. doi: 10.1063/1.868164. URL `http://link.aip.org/link/?PHF/6/2738/1`.

J. Tan and X. Yang. Physically-based fluid animation: A survey. *Science in China Series F: Information Sciences*, 52:723–740, 2009. ISSN 1009-2757. URL `http://dx.doi.org/10.1007/s11432-009-0091-z`. 10.1007/s11432-009-0091-z.

Z. Tan, K. Lim, and B. Khoo. An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model. *Journal of Computational Physics*, 225(1):1137 – 1158, 2007. ISSN 0021-9991. doi: DOI:10.1016/j.jcp.2007.01.019.

A. S. Tanenbaum. *Modern operating systems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992. ISBN 0-13-588187-0.

H.-Z. Tang, T. Tang, and P. Zhang. An adaptive mesh redistribution method for nonlinear hamilton–jacobi equations in two-and three-dimensions. *J. Comput. Phys.*, 188(2):543–572, 2003. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/S0021-9991(03)00192-X.

G. Taubin. Blic: Bi-level isosurface compression . In *IEEE Visualization*, 2002.

J. Tessendorf. Simulating ocean water. In *In SIGGRAPH Course Notes*, 2004.

S. Toledo. A survey of out-of-core algorithms in numerical linear algebra. *External memory algorithms*, pages 161–179, 1999.

C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface*, pages 26–34, 1998.

E. Turkel and A. Yefet. Absorbing pml boundary layers for wave-like equations. *Appl. Numer. Math.*, 27(4):533–557, 1998. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/S0168-9274(98)00026-9.

J. S. Vitter. External memory algorithms and data structures: dealing with massive data. *ACM Comput. Surv.*, 33(2):209–271, 2001. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/384192.384193.

T. Yabe, F. Xiao, and T. Utsumi. The constrained interpolation profile method for multiphase analysis. *J. Comput. Phys.*, 169(2):556–593, 2001. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2000.6625.

N. B. Zafar, D. Stephens, M. Larsson, R. Sakaguchi, M. Clive, R. Sampath, K. Museth, D. Blakey, B. Gazdik, and R. Thomas. Destroying la for "2012". In *SIGGRAPH '10: ACM SIGGRAPH 2010 Talks*, pages 1–1, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0394-1. doi: http://doi.acm.org/10.1145/1837026.1837059.

L. Zhao and A. Cangellaris. Gt-pml: generalized theory of perfectly matched layers and its application to the reflectionless truncation of finite-difference time-domain grids. *Microwave Theory and Techniques, IEEE Transactions on*, 44(12):2555 –2563, dec 1996. ISSN 0018-9480. doi: 10.1109/22.554601.

Y. Zhu and R. Bridson. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1186822.1073298.