Diss. ETH No. 16785

Meshless Lagrangian Methods for Physics-Based Animations of Solids and Fluids

A dissertation submitted to
ETH Zurich

for the degree of
Doctor of Sciences

presented by
Richard Keiser
Dipl. Ing. Inf. ETH Zurich, Switzerland
born January 20, 1978
citizen of Krummenau, SG, Switzerland

accepted on the recommendation of
Prof. Mark Pauly, ETH Zurich, Switzerland, examiner
Prof. Markus Gross, ETH Zurich, Switzerland, co-examiner
Prof. Leonidas J. Guibas, Stanford University, USA, co-examiner

2006

“O beatissime lector,
lava manus tuas et sic librum adprehende,
leniter folia turna,
longe a littera digito pone.
Quia qui nescit scribere,
putat hoc esse nullum laborem.
O quam gravis est scriptura:
oculos gravat,
renes frangit,
simul et omnia membra contristat.
Tria digita scribunt,
totus corpus laborat...”

Note of a writer, 8th century.

Abstract

Nature builds very complex systems of mutually interacting and changing materials. Computer graphics attempts to map these real world phenomena onto simplified physics-based models. The objective of this dissertation is to develop new methods for physics-based animation of interacting fluids and deformable objects, including melting and freezing, based on a unified meshless Lagrangian approach.

Discretizing and solving the equations of motion using a meshless particle-based system has several advantages compared to mesh-based Lagrangian and Eulerian methods. Material properties are simply advected with the particles and might change as a function of time, the particles' position, and properties of neighboring particles. For strong deformations, the spatial discretization can be efficiently adapted without the need for complicated remeshing. Furthermore, grid-based aliasing artifacts from the alignment with boundaries are avoided. A major drawback of meshless methods is the expensive computation of the particles' neighborhood needed for computing the shape functions. In this dissertation, the benefits of meshless collocation methods are explored for stably animating fluids and objects with arbitrary deformations.

In our framework, the domain of a model is represented by volumetric particles. We explore the Smoothed Particle Hydrodynamics method to stably and efficiently solve the continuum mechanics equations for simulating fluids and elasto-plastic objects including fracturing and contact handling. Based on a unified particle metaphor, we present a framework that combines solids and fluids, thus enabling to simulate a broad range of effects such as viscoelastic materials, melting solids, interactions between solids and fluids, and multiphase effects between liquid and air such as bubbles and foam. To improve the performance, we present a new multiresolution approach that adapts the discretization of the domain dynamically to the characteristics of the simulation, while reducing the overall complexity of the computations.

High quality animations require a high resolution surface embedded into the volumetric representation. We present new meshless methods for animating a point-sampled surface along with the particles. Our surface model exploits the advantages of both explicit and implicit surface representations. Geometrically complex surfaces of deformable solids are efficiently animated using a rigid motion invariant free-form deformation approach. Additionally, surface potential fields can guide the surface deformation such that topological changes are handled implicitly, making the model suitable also for fluid simulations as well as melting

and freezing.

In our research we found meshless Lagrangian methods for the volume and surface animation to be most suitable for the simulation of strongly deforming objects. The sampling of the physical domain is adapted in case of extreme deformations or topological changes using a simple and efficient resampling scheme. Similarly, a point-based surface handles topological changes and strong surface deformations by adapting the point-sampling dynamically without the need for maintaining the connectivity. We demonstrate the capability of our meshless animation framework on a large variety of examples, such as the physics-based animation of deformable objects ranging from stiff elastic to highly plastic, fracturing of both brittle and ductile material, contact handling of colliding deformable objects, high resolution animations of splashing fluids with bubbles, two-way coupling between solids and fluids, melting solids with highly detailed and textured surfaces, solidifying flowing liquids to elastic solids, and viscoelastic liquids with different physical characteristics.

Kurzfassung

In der Natur existieren sehr komplexe Systeme von gegenseitig interagierenden und sich stetig verändernden Materialien. In der Computer Grafik wird versucht diese Phänomene aus der reellen Welt auf ein simplifiziertes physikalisch basiertes Modell abzubilden. Das Ziel dieser Dissertation ist neue Methoden zu entwickeln um interagierende Flüssigkeiten und deformierbare Objekte, inklusive schmelzen und erstarren, nach physikalischen Gesetzen zu animieren, basierend auf einem netzlosen Lagrange-Ansatz.

Das Benutzen eines netzlosen partikelbasierten Systems zur Diskretisierung und Lösung von den Bewegungsgleichungen hat diverse Vorteile gegenüber netzbasierten Lagrange- und Euler-Methoden: Materialeigenschaften werden einfach mit den Partikeln mitgeführt und verändern sich in Abhängigkeit von Zeit, den Partikelpositionen, und Eigenschaften anderer Partikel. Bei sehr starken Deformationen kann die räumliche Diskretisierung effizient angepasst werden ohne ein komplizierte Neuvernetzung durchführen zu müssen. Zudem werden gitterbasierte Aliasing-Artefakte, die durch die Ausrichtung an Grenzflächen entstehen, vermieden. Ein Hauptnachteil von netzlosen Methoden ist die teure Berechnung der Partikelnachbarschaften, die benötigt werden um die *Shape*-Funktionen zu berechnen. In dieser Dissertation werden die Vorteile von netzlosen *Collocation*-Methoden erforscht für eine stabile Animation von Flüssigkeiten und Objekten mit beliebigen Deformationen.

In unserem System wird die Domäne des Modells mit volumetrischen Partikeln repräsentiert. Wir untersuchen die *Smoothed Particle Hydrodynamics* Methode um die Gleichungen der Kontinuummechanik für die Simulation von Flüssigkeiten und elasto-plastischen Objekten, inklusive Zerbrechen und Kontaktbehandlung, stabil und effizient zu lösen. Basierend auf einer einheitlichen Partikel-Metapher präsentieren wir ein System welches Festkörper und Flüssigkeiten kombiniert und damit die Simulation von einer breiten Palette von Effekten erlaubt, wie zum Beispiel viskoelastische Materialien, schmelzende Festkörper, Interaktion zwischen Festkörpern und Flüssigkeiten, und Multiphasen-Effekte zwischen Flüssigkeiten und Luft wie zum Beispiel Luftblasen und Schaum. Um die Berechnungsleistung zu verbessern präsentieren wir einen neuen *Multiresolution*-Ansatz, welcher dynamisch die Diskretisierung der Domäne dem Charakteristikum der Simulation anpasst, wobei gesamthaft die Komplexität der Berechnungen reduziert wird.

Qualitativ hochstehende Animationen benötigen eine hochaufgelöste Oberfläche die in die volumetrische Representation eingebettet ist. Wir stellen neue netzlose Methoden zur Animation von punktbasierten Oberflächen zusammen mit den Partikeln vor. Unser Oberflächenmodell nutzt die Vorteile sowohl von expliziten als auch impliziten Oberflächenrepräsentationen aus. Geometrisch komplexe Oberflächen von deformierbaren Festkörpern werden effizient animiert, wozu ein *Free-form*-Deformations-Ansatz benutzt wird der invariant ist für rigide Transformationen. Zusätzlich leiten Oberflächen-Potentialfelder die Oberflächendeformation so dass topologische Änderungen implizit behandelt werden, weshalb das Modell auch für Flüssigkeitssimulationen sowie für die Animation von Schmelzen und Erstarren geeignet ist.

Die Resultate unserer Forschung zeigen dass netzlose Lagrange-Methoden für die Volumen- und Oberflächen-Animation dann am Besten geeignet sind wenn die simulierten Materialien sehr stark deformiert werden. Das Sampling der physikalischen Domäne wird mittels einem einfachen und effizienten Resampling-Schema bei sehr starken Deformationen oder topologischen Änderungen angepasst. Auf eine ähnliche Weise passt sich die punktbasierte Oberfläche den topologischen Änderungen und starken Oberflächendeformationen an indem das Punktsampling dynamisch angepasst wird ohne dass dabei die Konnektivität erhalten werden muss. Wir demonstrieren die Fähigkeiten unseres netzlosen Animationssystems anhand verschiedener Beispiele, wie z.B. die physikalisch basierte Animation von deformierbaren Objekten mit Materialeigenschaften von steif-elastisch bis zu hoch plastisch, das Zerbrechen von sowohl spröden als auch plastischen Materialien, Kontaktbehandlung von kollidierenden deformierbaren Objekten, hoch aufgelöste Animationen von spritzenden Flüssigkeiten mit Luftblasen, gegenseitige Kupplung von Festkörpern und Flüssigkeiten, schmelzende Festkörper mit hoch detailierten und texturierten Oberflächen, verfestigen von fliessenden Flüssigkeiten zu elastischen Festkörpern, und viskoelastische Flüssigkeiten mit verschiedenen physikalischen Charakteristika.

Acknowledgments

Writing a dissertation, like a roller coaster, is filled with ups and downs and hardly ever goes the way you expect it to go. However, I was in the lucky position of working together with fantastic people. These people made even the downs seem fun and helped them to quickly change into ups again, yielding in a very satisfying, enjoyable and successful experience.

First of all I want to thank Mark Pauly. Mark was more a collaborator and friend than a supervisor. During this close collaboration, I was able to profit greatly from his knowledge and experience. My gratitude also goes to Markus Gross. Markus brought me into the exciting world of meshless physics-based animations and provided direction to my research with his visionary ideas. I also would like to thank Leonidas Guibas. Leo often revealed new ways of solving a problem and working in his lab in Stanford was a great experience. The last member of this strong team was Bart Adams. Whether working the nights through before SIGGRAPH, visiting the parks in California, or going for a couple of drinks in Leuven, we always had an enjoyable time!

Furthermore, I would like to thank the others with whom I collaborated and who made this dissertation possible, in particular Matthias Müller, Andy Nealen, and Dominique Gasser.

I would also like to thank everyone from the CGL and AGG in Zurich for their support and friendship. Especially, big thanks go to Martin Wicke and Miguel Otaduy for the fruitful discussions and the help I received from them, and for the great time we had exploring the bars of Zurich. Thanks to my former office mate Christian Sigg for providing the dissertation template.

Finally, I would like to express my gratitude to my family and friends for their endless support and motivation.

Contents

Abstract iii

Kurzfassung v

Acknowledgments vii

Notations xv

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 3
- 1.3 Animation Loop . . . 5
- 1.4 Outline . . . 5
- 1.5 Publications and Collaborations . . . 8

2 Foundations 11
- 2.1 Continuum Mechanics . . . 11
  - 2.1.1 Conservation Laws . . . 12
  - 2.1.2 Stress . . . 13
  - 2.1.3 Solid Mechanics . . . 14
  - 2.1.4 Fluid Mechanics . . . 15
  - 2.1.5 Viscoelastic Materials . . . 16
- 2.2 Physics Representations . . . 16
  - 2.2.1 Eulerian versus Lagrangian Methods . . . 17
  - 2.2.2 Mesh-based versus Meshless Methods . . . 18
- 2.3 Smoothed Particle Hydrodynamics . . . 18
  - 2.3.1 SPH Approximation Error . . . 20
  - 2.3.2 Corrective Smoothed Particle Method . . . 20
  - 2.3.3 Corrected Smoothed Particle Hydrodynamics . . . 21
  - 2.3.4 Moving Least-Squares Particle Hydrodynamics . . . 21
- 2.4 Surface Representations . . . 23
  - 2.4.1 Implicit and Explicit Representations . . . 24
  - 2.4.2 Projecting a Point onto an Implicit Surface . . . 26

2.5 Point-based Representation . . . 27
2.5.1 Implicit Surface from Points . . . 27

3 State of the Art 31
3.1 Meshless Lagrangian Methods . . . 31
3.1.1 Particle Systems . . . 31
3.1.2 Smoothed Particle Hydrodynamics . . . 35
3.2 Mesh-Based Lagrangian Methods . . . 36
3.2.1 The Finite Element Method . . . 36
3.2.2 The Finite Difference Method . . . 38
3.2.3 The Finite Volume Method . . . 38
3.2.4 The Boundary Element Method . . . 38
3.2.5 Mass-Spring Systems . . . 39
3.3 Eulerian and Semi-Lagrangian Methods . . . 40

4 Multiresolution Fluid Simulation 45
4.1 Introduction . . . 46
4.2 Related Work . . . 48
4.3 Fluid Model . . . 49
4.3.1 Initialization and SPH Force Approximation . . . 50
4.3.2 Kernels . . . 51
4.3.3 Color Field . . . 51
4.4 Multiphase SPH . . . 52
4.4.1 Air Generation . . . 52
4.4.2 Water-Air Interaction . . . 53
4.5 Multiresolution Particle System . . . 55
4.5.1 Virtual Particles . . . 56
4.5.2 Splitting and Merging . . . 58
4.6 Surface Extraction . . . 59
4.7 Results & Discussion . . . 60
4.7.1 Limitations . . . 62
4.8 Extensions & Future Work . . . 62
4.9 Summary . . . 64

5 Deformable Solid Simulation 65
5.1 Introduction . . . 66
5.2 Related Work . . . 67
5.2.1 Deformable Modeling . . . 67
5.2.2 Fracturing . . . 67
5.2.3 Contact Handling . . . 68
5.3 Continuum Mechanics Equations . . . 69
5.4 Elasticity Model . . . 70
5.4.1 Moving Least-Squares Approximation of $\nabla \mathbf{u}$ . . . 71
5.4.2 Elastic Force Computation . . . 73

5.4.3 Rigid Transformation of the Rest Shape . . . 74
5.4.4 Plasticity Model . . . 75
5.5 Surface Model . . . 75
5.5.1 Surface Animation . . . 76
5.5.2 Surface Refinement . . . 77
5.6 Fracture Model . . . 77
5.6.1 Introduction . . . 78
5.6.2 Modeling Discontinuities . . . 79
5.6.3 Fracture Surface Model . . . 81
5.6.4 Crack Initiation and Propagation . . . 82
5.6.5 Topology Control . . . 84
5.6.6 Fracture Control . . . 86
5.7 Volumetric Sampling . . . 87
5.8 Contact Model . . . 88
5.8.1 Overview . . . 89
5.8.2 Collision Detection . . . 89
5.8.3 Contact Surface . . . 91
5.8.4 Collision Response . . . 92
5.8.5 Contact Handling Pipeline . . . 95
5.9 Results & Discussion . . . 96
5.9.1 Limitations . . . 99
5.10 Extensions & Future Work . . . 100
5.11 Summary . . . 101

6 Solid-Fluid Simulation 103
6.1 Introduction . . . 104
6.2 Related Work . . . 104
6.2.1 Melting Objects and Viscoelastic Fluids . . . 105
6.2.2 Surface Extraction and Animation . . . 105
6.3 Physics Model . . . 106
6.3.1 Governing Equations . . . 106
6.3.2 Force Computations . . . 108
6.4 Particle Animation . . . 109
6.4.1 Deformation of the Rest Shape . . . 109
6.5 Melting and Solidifying . . . 110
6.6 Surface Animation . . . 112
6.6.1 Surfel Neighborhoods . . . 113
6.6.2 Surface Deformation . . . 114
6.6.3 Resampling . . . 118
6.6.4 Zombies . . . 119
6.6.5 Topological Changes . . . 119
6.6.6 Blending Between Solids and Fluids . . . 121
6.7 Results . . . 122
6.8 Limitations & Future Work . . . 127

6.9 Summary 128

7 Implementation **129**
7.1 Search Data Structures 129
7.1.1 Hash Grid 130
7.1.2 *k*d-Tree 131
7.1.3 Discussion 132
7.2 Time Integration 132
7.2.1 Leapfrog Integration 133
7.2.2 Discussion 134

8 Conclusion **135**
8.1 Summary 135
8.2 Discussion 137
8.2.1 Smoothed Particle Hydrodynamics 137
8.2.2 Point-based Representation 139
8.3 Future Work 140

Bibliography **143**

Copyrights **171**

Curriculum Vitae **173**

List of Figures

1.1 Animation loop 6

2.1 Visualization of the stress tensor 13
2.2 Eulerian, mesh-based and meshless Lagrangian representations 16

3.1 Simulation of a lava lamp using the SPH method 35
3.2 Artifacts of linear FEM under large rotational deformations 37
3.3 Cloth modeled using a mass-spring system 39
3.4 Melting bunny 42
3.5 A dripping viscoelastic fluid 43
3.6 Controlled fluid simulation using the lattice Boltzmann method 44

4.1 Eulerian vs. Lagrangian advection 47
4.2 Air generation according to colorfield 52
4.3 Multiresolution fluid simulation and rigid bodies-fluid interaction 54
4.4 Multiresolution coupling with virtual particles 57
4.5 Surface extraction from the color field and Delaunay triangulation 59
4.6 Multiphase fluid simulation of a splashing sphere 61
4.7 Particle-based rigid body collisions and interaction with fluid 61
4.8 Controlled fluid animation 63

5.1 Deformed objects consisting of volume and surface elements 69
5.2 Effect of Poisson's ratio 70
5.3 Particle-based approximation scheme 71
5.4 Rigid transformation of the rest shape 74
5.5 Real-time elastic and plastic deformations of Max Planck 75
5.6 Surface animation and refinement 76
5.7 Brittle fracture of a hollow stone sculpture 78
5.8 Ductile fracture of a bubble gum like material 79
5.9 Comparison of visibility and transparency method 80
5.10 Surfel clipping to create sharp creases and particle resampling 82
5.11 Front propagation and fracture surface sampling 83
5.12 Transparency weights for embedding surfels in the simulation domain . . . 84

5.13 Topological events during crack propagation 85
5.14 Controlled fracture 86
5.15 Volumetric sampling scheme 87
5.16 Dynamic resampling scheme 88
5.17 Contact handling pipeline 89
5.18 Contact surface computation 90
5.19 Contact surface example 92
5.20 Penalty and friction force computation 93
5.21 Plastic Max Planck models building a pile and falling apart again 94
5.22 Santa Claus riding the dragon. 95
5.23 Melting the Max Planck model 96
5.24 Animating a highly detailed octopus model 97
5.25 Newton's Cradle with stiff elastic spheres 98
5.26 Performance measurement for two colliding Max Planck models 99

6.1 Solidifying a fluid due to the contact with the frozen ground 107
6.2 Force computation pipeline 109
6.3 Melting two cubes through a funnel 111
6.4 Melting of a cube with sharp edges 113
6.5 Particle and surfel neighborhood scheme 114
6.6 Illustration of the guiding, smoothing and attracting force 116
6.7 Texture interpolating example using zombie particles 119
6.8 Surface splitting and merging 120
6.9 Melting of an elastic solid dropped onto a heated box 121
6.10 Pouring a fluid into a glass 122
6.11 Freezing a quicksilver fluid that is poured into a glass 123
6.12 Object melted through a funnel into a casting mold 124
6.13 Comparison of viscoleastic fluids 125

Notations

Abbreviations and Acronyms

1D	one–dimensional
2D	two–dimensional
3D	three–dimensional
AABB	Axis Aligned Bounding Box
ALE	Arbitrary Lagrangian-Eulerian
BEM	Boundary Element Method
CIP	Cubic Interpolated Propagation
CPU	Central Processing Unit
CSPM	Corrective Smoothed Particle Method
CSPH	Corrected Smoothed Particle Hydrodynamics
EMC	Extended Marching Cubes
ETH	Swiss Federal Institute of Technology
FDM	Finite Difference Method
FEM	Finite Element Method
FLIP	Fluid-Implicit-Particle method
FVM	Finite Volume Method
GPU	Graphics Processing Unit
LBM	Lattice Boltzmann Method
MC	Marching Cubes
MD	Molecular Dynamics
MLS	Moving Least–Squares
MLSPH	Moving Least–Squares Particle Hydrodynamics
MM	Meshless Method
MPS	Moving Particle Semi-implicit method
NS	Navier–Stokes
PCA	Principal Component Analysis
PIC	Particle-In-Cell method
RSPH	Regularized SPH
SPH	Smoothed Particle Hydrodynamics
VOF	Volume-Of-Fluid

Geometry

$\mathbf{x}$	point in $\mathbb{R}^3$ in world coordinates
$\mathbf{m}$	point in $\mathbb{R}^3$ in material coordinates
x,y,z	scalar coordinates: $\mathbf{x} = [x, y, z]^T$
$\mathbf{e}_i$	coordinate axis: $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$ are orthonormal and span $\mathbb{R}^3$
$\mathbf{r}_{ij}$	distance vector between two points $\mathbf{x}_i$ and $\mathbf{x}_j$: $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$

Operators & Norms

$\dot{\mathbf{x}}$	first time derivative: $\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t}$
$\ddot{\mathbf{x}}$	second time derivative: $\ddot{\mathbf{x}} = \frac{\partial^2 \mathbf{x}}{\partial^2 t}$
$_{,x}$,$_{,y}$,$_{,z}$	first spatial derivative: $_{,x} = \frac{\partial}{\partial x}$
∇	gradient: $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^T$
$\nabla\cdot$	divergence operator: $\nabla\cdot = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$
∇^2	laplace operator: $\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$
$\nabla_{\mathbf{u}}$	directional derivative (rate of change in direction $\mathbf{u}$)
$\frac{D}{Dt}$	material/substantial derivative: $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla$
$A * B$	convolution of two scalar functions A and B
$\|.\|$	2-norm (Euclidian norm)
$\|.\|$	absolute value of a scalar

Approximation Functions & Matrices

Φ	shape function
$\mathbf{b}$	polynomial basis
$\mathbf{a}$	vector of unknown coefficients
$A(.)$	scalar differentiable function
$\langle A(.) \rangle$	integral or discrete approximation of a scalar differentiable function
$\mathbf{M}$	moment matrix
$\mathbf{J}$	Jacobian matrix
$\mathbf{I}$	identity matrix
$\delta(\mathbf{r})$	delta function: $\delta(\mathbf{r}) = 1$ if $\|\mathbf{r}\| = 0$, 0 otherwise
e	least-squares error
h	support radius (smoothing length scale)
$\omega(\mathbf{r}, h)$	normalized kernel function with support h and depending on distance vector $\mathbf{r}$
λ	eigenvalue
$\mathbf{e}$	eigenvector

Implicit Surface Ψ

$F(\mathbf{x})$	implicit function, defining the surface as $\Psi = \{\mathbf{x} \in \mathbb{R}^3 \,\vert\, F(\mathbf{x}) = 0\}$
$\psi(\mathbf{x})$	projection of $\mathbf{x}$ onto Ψ
$\psi^{\text{orth}}(\mathbf{x})$	orthogonal projection of $\mathbf{x}$ onto Ψ
I	isovalue of a potential field

Object Γ

$\mathcal{S}$	surface
Ω	volume

Surfel s_i

$\mathbf{s}_i$	position in world coordinates
$\mathbf{t}^1_{s_i}, \mathbf{t}^2_{s_i}$	first and second tangent axes in world coordinates
$\mathbf{m}_{s_i}$	position in material coordinates
$\mathbf{t}^1_{m_i}, \mathbf{t}^2_{m_i}$	first and second orthogonal tangent axes in material coordinates
$\mathbf{n}_{s_i}$	normalized normal
$\mathcal{S}_i$	set of surfels in the neighborhood of s_i
$h_{\mathcal{S}_i}$	surfel support radius determining the neighborhood range for $\mathcal{S}_i$
$\mathcal{P}_i$	set of particles in the neighborhood of s_i
$h_{\mathcal{P}_i}$	particle support radius determining the neighborhood range of $\mathcal{P}_i$

Particle p_i

$\mathbf{p}_i$	position in world coordinates	[m]
$\mathbf{m}_i$	position in material coordinates	[m]
$\mathbf{u}_i$	displacement vector	[m]
$\mathbf{v}_i$	velocity	[m/s]
$\mathbf{a}_i$	acceleration	[m/s^2]
V_i	volume	[m^3]
m_i	mass	[kg]
ρ_i	density	[kg/m^3]
T_i	temperature	[°C]
h_i	support radius	[m]
P_i	pressure	[N/m^2]
$\mathbf{f}_i$	force acting on p_i	[N]
$\mathbf{f}_{ij}$	force exerted from particle p_j onto p_i	[N]
$\mathbf{f}_i^{\text{pressure}}$	pressure force acting on p_i	[N]
$\mathbf{f}_i^{\text{viscosity}}$	viscosity force acting on p_i	[N]
$\mathbf{f}_i^{\text{cohesion}}$	cohesion force acting on p_i	[N]
$\mathbf{f}_i^{\text{contact}}$	contact force acting on p_i	[N]
$\mathbf{f}_i^{\text{ext}}$	sum of external forces acting on p_i	[N]
$\mathbf{f}_i^{\text{total}}$	sum of all forces acting on p_i	[N]

Physics

t	time	[s]
Δt	simulation time step	[s]
ρ^0	rest density	$[kg/m^3]$
$\mathbf{g}$	gravity acceleration	$[m/s^2]$
$\mathbf{f}$	force	[N]
$\tilde{\mathbf{f}}$	force density vector field (force per volume)	$[N/m^3]$
φ	color field value	[1]
τ_φ	color field threshold for inserting a new air particle	[1]
E	Young's modulus	$[N/m^2]$
ν	Poisson's ratio	[1]
U	energy	[Nm]
$\tilde{U}$	energy density field (energy per volume)	$[N/m^2]$
ε^s	solid strain tensor without plasticity	[1]
σ^s	solid stress tensor without plasticity	$[N/m^2]$
ε^p	plastic strain tensor	[1]
ε^e	solid strain tensor with plasticity	[1]
σ^e	solid stress tensor with plasticity	$[N/m^2]$
σ^f	fluid stress tensor	$[N/m^2]$
k^{yield}	yield strength material constant	[1]
k^{creep}	plastic deformation material constant	$[s^{-1}]$
μ	material viscosity	$[Ns/m^2]$
k^{gas}	gas constant (material stiffness)	$[N/m^2]$
$k^{cohesion}$	cohesion constant	$[Nm^4/kg^2]$
T	temperature	[°C]
κ	thermal conductivity material coefficient	[W/(mK)]
k^{cap}	material heat capacity	[J/K]
k^{heat}	heat diffusion constant: $k^{heat} = \kappa / k^{cap}$	$[(ms)^{-1}]$

Surface Animation

Θ^{guide}	guiding potential
$\mathbf{f}^{guide}$	guiding force
k^{guide}	guiding force influence constant
$\Theta^{attract}$	attracting potential
$\mathbf{f}^{attract}$	attracting force
$k^{attract}$	attracting force influence constant
Θ^{smooth}	smoothing potential
$\mathbf{f}^{smooth}$	smoothing force
k^{smooth}	smoothing force influence constant
Θ^{repel}	repulsion potential
$\mathbf{f}^{repel}$	repulsion force
k^{repel}	repulsion force influence constant

Fracture

$\mathbf{c}$	crack node on a fracture surface, consisting of two surfels with opposing normals
$\mathbf{d}$	propagation vector at a crack node
$\mathbf{t}$	tangent vector to the crack front
k^{prop}	propagation speed factor of the crack front
k^{opacity}	opacity of a crack defining the visibility between two particles
k^{hist}	factor for adjusting the propagation direction for controlled fracture simulation

Contact Handling

$\mathcal{B}$	object bounding box
s_{c}	surfel on the contact surface
$\mathbf{s}_{\text{c}}$	position of s_{c} in world coordinates
c	a contact node consisting of two contact surfels with opposing normals
$\mathbf{c}$	position of c in world coordinates
$\mathbf{n}_c$	normalized normal direction of c
ρ_c	local contact node density at $\mathbf{c}$
h_c	particle support radius of a contact node, defines the particle neighborhood of c
$\mathbf{d}_c$	penetration direction of two objects at c
$\mathbf{v}_c$	relative velocity of two intersecting objects at $\mathbf{c}$
$\mathcal{S}_{\text{c}}$	contact surface
Ω_{c}	intersection volume between two objects
$\mathbf{f}_c^{\text{pen}}$	penalty force acting on c
k^{pen}	penalty force constant
$\mathbf{f}_c^{\text{fri}}$	friction force acting on c
μ^{fri}	frictional material coefficient
k^{damp}	surface force damping constant
$\mathbf{f}_c^{\text{surface}}$	total force acting on c
$\mathbf{f}_{i,c}^{\text{contact}}$	force exerted from contact node c onto a particle p_i
$\mathbf{f}_i^{\text{contact}}$	total contact force acting on a particle p_i

Chapter 1

Introduction

This chapter gives an overview and motivation of this dissertation and our contributions. We briefly discuss the advantages of using a meshless approach to solid-fluid simulation and the objectives of our work in Section 1.1. The major contributions to the state of the art in physics-based computer animation of solids and fluids are listed in Section 1.2. Section 1.3 describes a high-level view of the animation loop. An outline of the structure of the dissertation is given in Section 1.4. We provide a list of publications that resulted from our work on physics-based animation and point-based representations in Section 1.5, and acknowledge our collaborators.

1.1 Motivation

Realistic animation of physical phenomena has gained increasing importance in many fields of computer graphics, including virtual surgery and the game and special effects industries. Typical examples include the simulation of rigid and deformable objects, and fluids. Furthermore, in nature solids and fluids are most often coupled. Thus, for realistic simulations the interaction between solids and fluids needs to be modeled. This also includes phase transitions between solids and liquids, i.e., melting and freezing.

A variety of methods exists for solving the equations of motion in physics-based animation. However, most methods are suitable for simulating either fluids or deformable bodies, but not both. Generally, these methods fall into two categories: Lagrangian and Eulerian methods. *Mesh-based Lagrangian* methods such as the finite element method and mass-spring systems are most often used to simulate deformable solids, whereas *Eulerian* methods using finite differences are state of the art in fluid simulation in computer graphics. This is due to the complementary advantages and drawbacks of these two approaches. In Lagrangian methods, the mesh moves with the material and therefore automatically adapts to the dynamics of the simulation, for instance when compressing or stretching an

object. Furthermore, due to the material description, the boundary of an object is explicitly defined. This not only simplifies the animation of the object's surface, but also the interaction with other objects and the enforcement of essential boundary conditions. Furthermore, it enables the simulation of complex effects such as fracturing. In contrast, Eulerian methods look at the evolution of material at fixed points in space. Thus, changes of topology are captured implicitly and even extreme deformations can be handled stably and efficiently without the need for remeshing of the domain or taking care of degenerated mesh elements. The biggest disadvantage of Eulerian methods is that the boundary of a fluid is not explicitly defined, and therefore the surface needs to be advected with the material flow. Furthermore, correct handling of boundaries and interacting objects that do not align with the Eulerian mesh is challenging.

Eulerian-based fluids and Lagrangian-based solids are usually coupled by rasterizing the solid onto the Eulerian mesh to prescribe velocity boundary conditions on the fluid and pressure forces on the solid [CMT04, GSLF05, LIGF06]. However, this is problematic because the discretization of the solid introduces errors and it is difficult to guarantee that fluid does not leak through [GSLF05]. Furthermore, the transition from a solid into a fluid while melting is very challenging because it requires changing continuously from the Lagrangian to the Eulerian representation [LIGF06].

Meshless Lagrangian methods, so-called *particle systems*, exploit advantages of both mesh-based Lagrangian and Eulerian methods. As a Lagrangian method, material properties are advected simply with the particles. The boundary of the physical domain is defined explicitly by the particles. Moreover, meshless methods can handle topological changes without the need for remeshing. A drawback of meshless collocation methods is that the particle neighborhood needs to be recomputed in every time step for adapting the shape functions, which is often the computationally most expensive part in a simulation. Furthermore, the non-interpolating shape functions render the enforcement of essential boundary conditions difficult.

To solve the aforementioned problem of coupling fluids and solids, we investigate meshless Lagrangian methods based on Smoothed Particle Hydrodynamics (SPH) for both deformable objects and fluids, and their combination. Treating both solids and fluids with the same representation simplifies handling their interaction, and renders freezing and melting materials possible. Furthermore, we extend the meshless collocation methods to incorporate fracturing, and develop a multiresolution approach that dynamically adapts the resolution of the particle system to the characteristics of the simulation.

Particle-based methods require the definition or extraction of an implicit or explicit surface from the particles. While an implicit definition of the surface defined by the particles can handle topological changes automatically by construction, the surface resolution depends on the (usually coarse) resolution of the particles. Hence, an explicit surface representation is more suitable for the simulation of highly detailed objects. On the other hand, fluid surfaces often rapidly change

their topology, which is hard to capture using an explicit approach. In our attempt to unify solid-fluid animations, we investigate point-based surface reconstruction methods that exploit advantages of both implicit and explicit representations. They are thus suitable for the animation of both deformable solids with highly detailed surfaces and fluids with smooth surfaces that change their topology frequently, as well as for the (local) transition between a solid and fluid surface while melting or freezing.

For engineering purposes, the focus of simulation methods is on accuracy. A new method is usually tested and compared with examples whose solutions can be either computed analytically or are known from experiments. However, analytical solutions often only exists for very simple test cases, and qualitative comparisons are usually restricted to 1D or 2D simulations. In computer graphics we aim to simulate highly complex phenomena in 3D that would be prohibitive to model by hand, for instance for virtual simulators, feature films, and games. Furthermore, in many applications the user interacts with the animated objects. Thus, the focus is on stability, physical plausibility, and speed, where usually quite big time steps are used. At the same time, the system designer often wants to be able to control the simulation, or even aims to achieve non-physical behavior, for instance, when animating a fluid character [SY05, TKPR06] such as the tar monster in the Scooby Doo 2 feature film [WH04]. To fulfill these requirements, accuracy and correctness is often traded off with computational complexity.

1.2 Contributions

In this dissertation we present meshless Lagrangian methods for the physics-based animation of both fluids and deformable objects that fracture, as well as the interaction between fluids and solids including melting and freezing, based on a unified particle metaphor. The major contributions are:

- **a multiphase fluid approach for simulating two-way coupling between liquids and air.** Air particles are generated dynamically at the liquid interface. Surface tension is simulated by modeling cohesion at the liquid-air interface. Trapped air particles turn into bubbles. Our multiphase method enables simulation of arbitrary fluid-fluid interaction. By sampling a solid with particles and treating it as a (rigid) fluid, the same interaction model as for fluid-fluid simulation can be used to simulate fluid-solid interactions.
- **a meshless collocation model based on continuum mechanics for the animation of elastic, plastic and melting objects.** We derive elastic forces in accordance with a linear displacement, constant strain approach. In contrast to most standard meshless approaches, which require solving complex integrals numerically, our method yields simple explicit equations which are easy to code and result in fast and stable animations.

- **a free-form deformation approach for animating the surface along with the particles**. A continuous displacement field is computed from the particles. This field is exploited to efficiently deform a point-sampled surface, where the deformation is invariant under rigid body motions. To prevent surface distortions for large deformations, a resampling scheme is applied.
- **a method for simulating elastic and plastic materials that fracture**. Fracture surfaces are dynamically created and maintained by continuously adding surface samples during crack propagation. Dynamic resampling adapts the particle sampling resolution to handle fracturing and large deformations. When new crack surfaces are created or the sampling of the domain changes, the shape functions adapt dynamically. Our method can handle complex topological events associated with multiple branching and merging cracks.
- **a collision detection and response algorithm for Lagrangian animations of deformable bodies, where both the volume and the surface representation are meshless.** The method stably resolves collisions for stiff elastic as well as highly deformable or plastic models. During collisions, it deforms the point-sampled surface and exerts forces on the volumetric particles. The decoupling of collision handling and deformations yields plausible collision simulations at interactive speed.
- **a unified approach for solid-fluid simulation including melting, freezing, and viscoelastic materials.** The equations of motion for solids and fluids are unified such that the same solver can be used for simulating both fluids and solids as well as viscoelastic materials. Furthermore, the physical material characteristics can change locally, which can be used to simulate effects such as melting and freezing.
- **a hybrid implicit-explicit surface generation approach that dynamically constructs a point-sampled surface wrapped around the particles**. Potentials are defined which guide the surface deformation. This surface representation enables modeling the fine surface detail required for solids, as well as the smooth surfaces of fluids, and the transition from detailed solid surfaces to smooth fluid surfaces during melting. Topological changes are incorporated in a lightweight and efficient manner. Furthermore, blending artifacts are avoided that typically arise when handling topological changes using implicit functions.
- **a multiresolution approach which automatically adapts the resolution to the simulation characteristics**. We introduce virtual particles to achieve a consistent coupling of different particle resolutions and use these particles to locally change the resolution in a consistent fashion. The method is simple and has only a very small overhead, resulting in a performance gain up to a factor of six in our examples. Furthermore, the approach is versatile in that it can be applied to most particle methods.

1.3 Animation Loop

Figure 1.1 gives a high-level overview of our animation loop. It consists of three loosely coupled parts, which are executed in the following order: Animation of the object's volume, animation of the surface, and contact handling of the surface with response forces influencing the physics simulation. Before the simulation starts, the physical domain is sampled with particles and the particles' properties are initialized as described in Sections 4.3.1 and 5.7 for fluids and solids, respectively. Alternatively, the particles are generated randomly in a cylinder by a source with a given initial velocity.

The physics animation starts with updating the rest shape of an object and recomputing the neighborhood of particles in case the rest shape of an object has changed. From the neighborhood, the density of a particle is approximated using SPH, which is then used for all other SPH approximations. Thus, this is a two-pass algorithm, which requires storing the particles' neighborhood. In a next step, strain and stress are computed, which are then used to derive the elastic forces. Additionally, fluid forces, such as pressure and viscosity forces, and forces between fluids and solids are computed. The move of the particles in time due to the applied forces is computed using the explicit leapfrog time integration scheme.In a next step, the computed stress is used to determine whether and where new cracks start or how existing cracks propagate. Changes in the dynamic or due to fracturing can make a local dynamic resampling of the domain necessary.

After a physics' step, the surface is animated by exploiting the continuously defined displacement field which is computed from the displacement of the particles. The surface is subsequently deformed according to potential fields to adapt it to the characteristics of the physics.

Finally, the deformed surface is used to detect collisions between deformable objects. Penalty forces are computed per surface element and distributed to the particles. These forces are then added to the force computation during the physics animation and thus influence the dynamic behavior of the objects.

1.4 Outline

The dissertation is organized as follows:

- **Chapter 2** introduces the fundamental concepts and definitions used in this dissertation. In Section 2.1, the governing equations and terms of continuum mechanics are introduced, including a discussion of solid and fluid mechanics, and viscoelastic materials. In Section 2.2 we discuss and compare three main classes of numerical approximation techniques, namely Eulerian, mesh-based and meshless Lagrangian methods. We then introduce Smoothed Particle Hydrodynamics and variants of it, which are used in this dissertation as meshless collocation methods (Section 2.3). Similarly to the

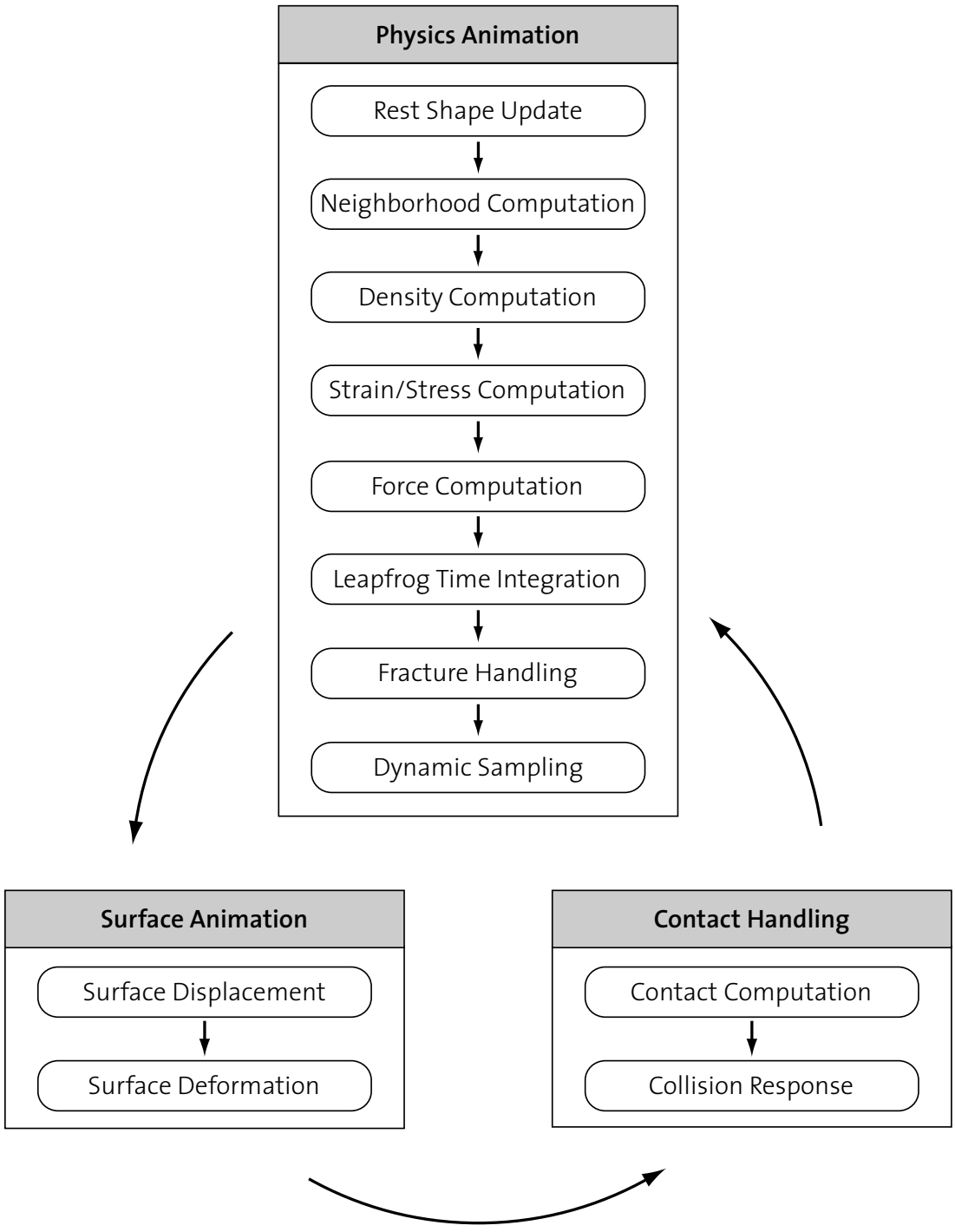


Figure 1.1: Animation loop. In the physics stage, the deformation of the volume of an object or fluid is computed and fracturing is handled. In the second stage, the surface is animated along with the volume and adapted to the particles. In the third stage, surface penetration is handled and response forces are computed that influence the physics animation.

different classes of numerical methods used to solve the physics equations, two main classes of surface representations exists, namely explicit and implicit representations, which are discussed in Section 2.4. We introduce point-sampled surfaces and argue that this representation combines the advantages of explicit and implicit representations (Section 2.5).

- **Chapter 3** presents the state of the art in physics-based deformable object and fluid simulation in computer graphics. We provide a background on particle-based systems (Section 3.1) and briefly discuss the most relevant work in mesh-based Lagrangian (Section 3.2), and Eulerian and semi-Lagrangrian methods (Section 3.3).

- **Chapter 4** presents a new multiresolution particle-based method for adapting the particle resolution to the characteristics of a fluid simulation and a multiphase approach for two-way air-water and fluid-solid coupling. Section 4.3 introduces our fluid model based on Smoothed Particle Hydrodynamics. This model is extended in Section 4.4 to support the simulation of multiphase fluids and interaction between fluids and rigid bodies. We then show in Section 4.5 how to decrease the computational complexity for fluid simulation using a multiresolution technique based on virtual particles. These particles yield a consistent coupling between different resolution levels and dynamic resolution adaption. Finally, we briefly discuss in Section 4.6 a new Lagrangian technique for extracting the surface.

- **Chapter 5** presents a framework for animating deformable objects including fracturing and contact handling. First, we derive the governing continuum mechanics equations and resulting forces based on a given deformation field (Section 5.3). We present in Section 5.4 our discrete elasticity model based on a moving least-squares approximation of the spatial derivatives of the displacement field from neighboring particles. Our surface model is introduced in Section 5.5. It enables fast animation of an embedded, highly detailed point-sampled surface with dynamic resampling to maintain a high quality surface. The elasticity and surface model is extended in Section 5.6 to support fracturing of both brittle and ductile material by adapting the shape functions and dynamic creation of the fracture surfaces. Dynamic adaption of the volumetric sampling to guarantee a stable simulation also for strong deformations and fracturing is discussed in Section 5.7. In Section 5.8 we introduce a new contact handling method for Lagrangian animated deformable objects with point-sampled surfaces, where the collision handling and deformation is decoupled to achieve efficient and stable simulations.

- **Chapter 6** combines and extends the techniques described in the previous chapters to unify the simulation of fluids and deformable solids, thus enabling the simulation of viscoelastic materials and effects such as melting and freezing. We present a new approach to surface extraction that can

handle the animation of objects with highly detailed surfaces, fluids with smooth surfaces, and melting deformable objects to fluids. We first present our physics model in Section 6.3 and show how this model can be used to simulate melting and solidifying by simple adapting the physical parameters (Section 6.5). In Section 6.6 we propose a new surface model based on potential fields that guide the deformation of the point-sampled surface.

- **Chapter 7** provides implementation details of the search data structures used in this dissertation, namely *k*d-tree and hash grids, and discusses the applied time integration schemes.
- **Chapter 8** concludes the dissertation with a summary of the previous chapters, a discussion of advantages and drawbacks of the applied approaches, and an outlook on future work.

1.5 Publications and Collaborations

The work presented in this dissertation is the result of international collaborations and yielded the following publications:

- The state of the art report given in Chapter 3 draws from a presentation at Eurographics 2005 and a publication in the Computer Graphics Forum Journal [NMK+06]. Collaborators are Andrew Nealen from the Technical University Berlin, Germany, Matthias Müller from AGEIA/NovodeX, Switzerland, Eddy Boxermann from the University of British Columbia, Canada, and Mark Carlson from DNA Productions Inc., USA.
- The multiresolution fluid framework including multiphase fluid simulation presented in Chapter 4 is on-going work and was published as a technical report [KAG+06]. Collaborators are Bart Adams and Philip Dutré from the Katholieke Universiteit Leuven, Belgium, Leonidas J. Guibas from Stanford University, USA, and Mark Pauly from ETH Zurich, Switzerland.
- Chapter 5 resulted in the following publications:
  - The basic framework for deforming elasto-plastic objects was published and presented at the ACM SIGGRAPH/Eurographics 2004 Symposium on Computer Animation in Grenoble, France [MKN+04]. Matthias Müller from AGEIA/NovodeX, Switzerland, derived the continuum mechanics equations presented in Section 5.3. Other collaborators are Andrew Nealen and Marc Alexa from the Technical University Berlin, Germany, and Mark Pauly and Markus Gross from ETH Zurich, Switzerland.
  - The free-form surface deformation approach used in the work described above draws from a publication in the ACM Transactions on

Graphics Journal [PKKG03] that was presented at ACM SIGGRAPH 2003 in San Diego, USA. Collaborators are Leif P. Kobbelt from the Rheinisch-Westfälische Technische Hochschule Aachen, Germany, and Mark Pauly and Markus Gross from ETH Zurich, Switzerland.

  - The extension for fracturing was presented at ACM SIGGRAPH 2005 in Los Angeles, USA, and published in the ACM Transactions on Graphics Journal [PKA+05]. Collaborators are Bart Adams and Philip Dutré from the Katholieke Universiteit Leuven, Belgium, Leonidas J. Guibas from Stanford University, USA, and Mark Pauly and Markus Gross from ETH Zurich, Switzerland.
  - Contact handling of deformable point-based objects was published and presented at the conference of Vision, Modeling and Visualization (VMV) 2004 in Stanford, USA [KMH+04]. Collaborators are Matthias Müller and Bruno Heidelberger from AGEIA/NovodeX, Switzerland, Matthias Teschner from University of Freiburg, Germany, and Markus Gross from ETH Zurich, Switzerland.

- The unified approach for solid-fluid simulations presented in Chapter 6 was published and presented at the Eurographics Symposium on Point-Based Graphics 2005, New York, USA [KAG+05]. Collaborators are Bart Adams and Philip Dutré from the Katholieke Universiteit Leuven, Belgium, and Dominique Gasser, Paolo Bazzi and Markus Gross from ETH Zurich, Switzerland.

Further publications that are related to this dissertation but not thoroughly discussed in here are:

- *Detail-preserving fluid control* [TKPR06], which will be presented and published at the ACM SIGGRAPH/Eurographics 2006 Symposium on Computer Animation, Vienna, Austria, in collaboration with Nils Thürey and Ulrich Rüde from the Friedrich-Alexander-Unitersity Erlangen-Nürnberg, Germany, and Mark Pauly from ETH Zurich, Switzerland.
- *Efficient raytracing of deforming point-sampled surfaces* [AKP+05], presented at Eurographics 2005, Dublin, Ireland, and published in the Computer Graphics Forum Journal, in collaboration with Bart Adams and Philip Dutré from the Katholieke Universiteit Leuven, Belgium, Leonidas J. Guibas from Stanford University, USA, and Mark Pauly and Markus Gross from ETH Zurich, Switzerland.
- *Particle-based fluid-fluid interaction* [MSKG05], presented and published at the ACM SIGGRAPH/Eurographics 2005 Symposium on Computer Animation, Los Angeles, USA, in collaboration with Matthias Müller from AGEIA/NovodeX, Switzerland, Barbara Solenthaler from the University of Zurich, Switzerland, and Markus Gross from ETH Zurich, Switzerland.

- *Consistent penetration depth estimation for deformable collision response* [HTK+04], published and presented at the conference of Vision, Modeling and Visualziation (VMV) 2004, Stanford, USA, in collaboration with Matthias Müller and Bruno Heidelberger from AGEIA/NovodeX, Switzerland, Matthias Teschner from University of Freiburg, Germany, and Markus Gross from ETH Zurich, Switzerland.
- *Post-processing of scanned 3D surface data* [WPK+04], published and presented at the Eurographics Symposium on Point-Based Graphics 2004, Zurich, Switzerland, in collaboration with Tim Weyrich, Mark Pauly, Simon Heinzle and Markus Gross from ETH Zurich, Switzerland, and Sascha Scandella from Cyfex, Switzerland.
- *Multi-scale feature extraction on point-sampled surfaces* [PKG03], was presented at Eurographics 2003, Granada, Spain, and published in the Computer Graphics Forum Journal, in collaboration with Mark Pauly and Markus Gross from ETH Zurich, Switzerland.
- *Collision detection and response for interactive editing of point-sampled models* [Kei03], Master thesis 2003, ETH Zurich, Switzerland.

Chapter 2

Foundations

This chapter introduces the basics of surface and physics representations and the fundamental equations of continuum mechanics that are relevant for this dissertation. Section 2.1 summarizes the conservation laws in physics, introduces the fundamental concept of stress and strain, and briefly discusses solid and fluid mechanics and viscoelastic materials. Different viewpoints on how the dynamics of material can be computed are given in Section 2.2, where mesh-based and meshless Lagrangian methods are compared. Details about the meshless approximation methods used in this dissertation are provided in Section 2.3. We then discuss and compare implicit and explicit surface representations in Section 2.4 and describe common methods for projecting a point onto an implicit surface. Section 2.5 introduces point-sampled surfaces used in this dissertation as an explicit surface representation and the underlying implicit representation.

2.1 Continuum Mechanics

In this section we describe the basic equations and introduce the most important terms of continuum mechanics, see [Chu96, Fun94, LRK93, Liu02b, BW97, BLM00] for detailed introductions and [Lov27] for a nice review of the history of elasticity theory. In *continuum theory*, matter is regarded as a continuum, i.e., indefinitely divisible, where small scale effects coming from molecular, atomic or sub-atomic interrelations are neglected. Thus, physical quantities such as energy and momentum can be handled in the infinitesimal limit, resulting in differential equations. Continuum mechanics is divided into solid mechanics and fluid mechanics. The most important difference between solids and fluids is that solids have a *rest shape* defined. *Elastic forces* counteract deformations from this rest shape. On the other hand, fluids flow because they cannot resist shear stress. Viscoelastic materials exhibit the characteristics of both fluid and solid. We will use the theory of continuum mechanics and of the different materials described below for the numerical simulation of fluids (Section 4.3), for deriving the (dis-

crete) elastic forces (Sections 5.3 and 5.4), and for deriving a model for animating viscoelastic materials (Section 6.3), and melting and freezing (Section 6.5).

We will first summarize the conservation laws of physics that are most relevant for our framework, and then describe an important mechanics quantity, the stress tensor (Section 2.1.2). We will then have a closer look at solid mechanics (Section 2.1.3), fluid mechanics (Section 2.1.4), and viscoelastic materials (Section 2.1.2). In this section we only consider classical mechanics (also called *Newtonian mechanics*), but no quantum mechanics.

2.1.1 Conservation Laws

The *constitutive equations* describe the relation between physical quantities specific to the material (see Sections 2.1.3 and 2.1.4), whereas the *conservation laws* of physics, which state that a physical quantity in an isolated system does not change, are valid for all materials. Important quantities that need to be conserved in our system are mass, energy and momentum. We will describe the governing equations from a *Lagrangian viewpoint* (also called *material* or *reference description*, see also Section 2.2.1), i.e., by following a particle that represents an infinitesimal volume of material.

The conservation of momentum is represented by *Newton's second law*, which describes the trajectory $\mathbf{p} = \mathbf{p}(t)$ of a particle as a function of time, i.e.,

$$\ddot{\mathbf{p}} = f(\dot{\mathbf{p}}, \mathbf{p}, t), \tag{2.1}$$

where $\ddot{\mathbf{p}}$ and $\dot{\mathbf{p}}$ are the second and first time derivatives of $\mathbf{p}$, and $f()$ is a function depending on the physical model. This second order differential equation can be written as a coupled set of ordinary differential equations

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{2.2}$$
$$\dot{\mathbf{v}} = f(\mathbf{v}, \mathbf{p}, t), \tag{2.3}$$

where $\mathbf{v}$ is the particle's velocity.

When following a particle, its volume dV and its density ρ may change, but its total mass $m = \rho dV$ will remain unchanged. The *continuity equation* represents the conservation of mass

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}, \tag{2.4}$$

where the *material derivative* (also called substantive, Lagrangian, or advective derivative)

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla \tag{2.5}$$

describes the rate of change of a physical quantity of a particle in time and $\nabla \cdot \mathbf{v}$ is the *divergence* of the velocity field. Note that the material derivative relates

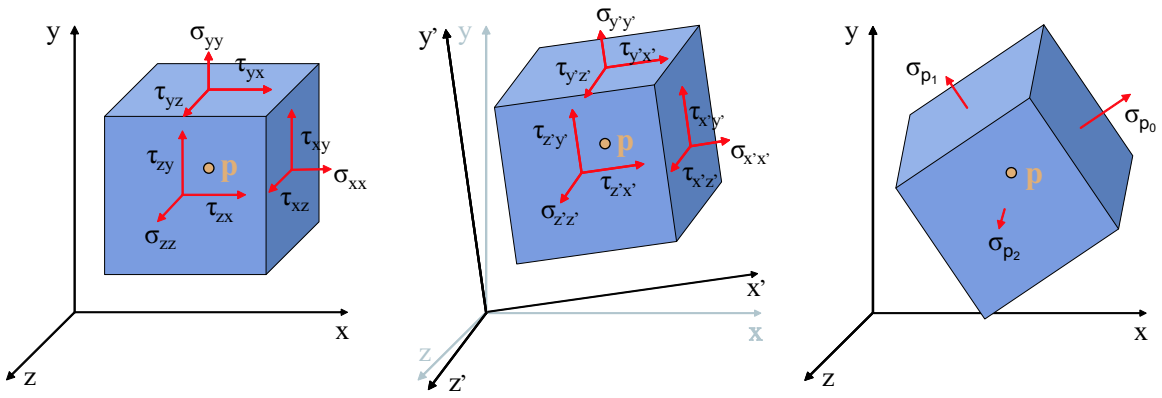


Figure 2.1: Left: visualization of the components of the symmetric stress tensor at **p**. Middle: the same state of stress is represented by a different set of components if the coordinate axes are rotated. Right: principal stresses.

the Lagrangian description to the *Eulerian description* (also known as *spatial description*) that describes the change of physical quantities at fixed points in space (known as *spatial coordinates*).

For an *incompressible material* the density of a particle does not change, i.e., $\frac{D\rho}{Dt} = 0$. Thus, mass conservation reduces to preserving a divergence free velocity field

$$\nabla \cdot \mathbf{v} = \mathbf{0}. \tag{2.6}$$

The conservation of energy represents the first law of thermodynamics, which states that the change in internal energy is equal to the heat added to the system minus the work done by the system.

2.1.2 Stress

In physics, the measurement of force per unit area is called stress. It is described in 3D by a symmetric 3×3 tensor

$$\sigma = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix}. \tag{2.7}$$

Both fluids and solids deform due to stresses, where the force $d\mathbf{f}$ acting on a differential surface element with area dA and normal vector $\mathbf{n}_A$ is given by

$$\frac{d\mathbf{f}}{dA} = \sigma \cdot \mathbf{n}_A. \tag{2.8}$$

A *tensor* is a multidimensional array that is independent of any chosen frame of reference, i.e., coordinate system, cf. Figure 2.1. The 3×3 stress tensor has *rank* 2,

which means that it has a magnitude and two directions associated with it (vectors have rank 1 (magnitude + direction) and scalars have rank 0 (only magnitude)). The two directions are called *normal stress* (the force component acting in direction of $\mathbf{n}_A$, denoted by σ_{ii}) and *shear stress* (the parallel components of the stress, denoted by τ_{ij}). The shear stress can be further decomposed into two orthogonal force components in the plane of $\mathbf{n}_A$. At a given point, it is always possible to find three orthogonal axes such that the shear stresses in the plane orthogonal to these axes vanish, cf. Figure 2.1. These planes are called *principal planes* and the stresses along the three axes are called *principal stresses*. They can be computed using principal component analysis (PCA) of the stress tensor, where the eigenvalues give the values of the principal stresses and the corresponding eigenvectors the direction.

The force $d\mathbf{f}$ acting on an infinitesimal volumetric element dV(a particle) due to internal stresses can be computed from Equation (2.8) as (for a derivation see e.g. [GP06])

$$\frac{d\mathbf{f}}{dV} = \nabla \cdot \sigma. \tag{2.9}$$

Applying Newton's second law $m\ddot{\mathbf{p}} = \mathbf{f}$ and dividing both sides with the volume $dV = m/\rho$ of a particle, the momentum equation can be written as

$$\rho\ddot{\mathbf{p}} = \nabla \cdot \sigma + \tilde{\mathbf{f}}^{\text{ext}}, \tag{2.10}$$

where $\tilde{\mathbf{f}}^{\text{ext}}$ is an external force density vector field (force per unit volume).

2.1.3 Solid Mechanics

A solid has a rest shape. If stresses are applied, the material deforms. An elastic solid has restoring forces for both normal and shear stress and will therefore return to its rest shape when no stresses are applied anymore. If the shear stress exceeds a materials *elastic range*, i.e., the stress is higher then the *yield strength* k^{yield} of a material, the deformations are non-reversible and material becomes (locally) *plastic*. *Brittle* materials fracture if the normal stress is too high, whereas *ductile* material undergo first plastic deformations under normal stress before they fracture. The simulation of brittle and ductile fracture will be described in Section 5.6, see also Figures 5.7 and 5.8.

The amount of deviation from the rest shape is described by the *strain tensor* ε. For most solids the strain is about proportional to the stress throughout their elastic range. For these *linear-elastic materials* (or *Hookean* materials), *Hooke's law* applies

$$\sigma^{\text{s}} = \mathbf{C}\,\varepsilon, \tag{2.11}$$

where σ^{s} is the solid stress and $\mathbf{C}$ is a rank four constant tensor that depends on Young's modulus E and Poisson's ratio ν. *Young's modulus* (also known as *modulus of elasticity*, *elastic modulus* or *tensile modulus*) determines the stiffness

of a given material. *Poisson's ratio* is a measurement for the incompressibility of a material, where $\nu = 0.5$ for a perfectly incompressible material and $0 \leq \nu < 0.5$ otherwise, cf. Figure 5.2. Because Hooke's law in combination with a linear strain model yields a linear equation system that can be solved efficiently, it is common in computer graphics to apply Hooke's law even for non-Hookean material such as rubber. In our physics model we derive $\mathbf{C}$ for isotropic Hookean materials, but use a non-linear strain model that handles rotations correctly, see Section 5.3 for details.

2.1.4 Fluid Mechanics

A fluid (a gas or a liquid) is different from a solid in that it is unable to sustain shearing stresses without continuously deforming. The amount of resistance of shearing stresses due to friction is called *viscosity*. Fluids are called *linearly viscous* or *Newtonian* if the viscous stress σ^{viscous} depends linearly on the rate of strain $\dot{\varepsilon}$

$$\sigma^{\text{viscous}} = \mu\,\dot{\varepsilon}, \tag{2.12}$$

where the viscosity μ is a material constant. Most fluids have a constant viscosity over a wide range of shear rates. Fluids are called *non-Newtonian*, if their viscosity changes with the strain rate (see also the next section).

The isotropic stress σ^{pressure} in normal direction is defined by the scalar P known as *hydrostatic pressure*. The total fluid stress is then the sum of the isotropic pressure stress and the viscous stress

$$\sigma^{\text{f}} = -P\mathbf{I} + \sigma^{\text{viscous}}, \tag{2.13}$$

where $\mathbf{I}$ is the identity matrix.

The Navier-Stokes equations [LL87] are a set of partial differential equations that state the conservation of momentum (Equation (2.14)), mass (Equation (2.15)) and energy (Equation (2.16)):

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla\cdot\sigma^{\text{f}} + \tilde{\mathbf{f}}^{\text{ext}}, \tag{2.14}$$

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}, \tag{2.15}$$

$$\rho\frac{DU}{Dt} = \sigma^{\text{f}}\cdot\nabla\mathbf{v} - \nabla\cdot\mathbf{q}, \tag{2.16}$$

where $\tilde{\mathbf{f}}^{\text{ext}}$ is an external force density vector field, U the internal energy and $\mathbf{q}$ the heat flux. In Section 4.3 we will describe how to solve these equations numerically using the Smoothed Particle Hydrodynamics method.

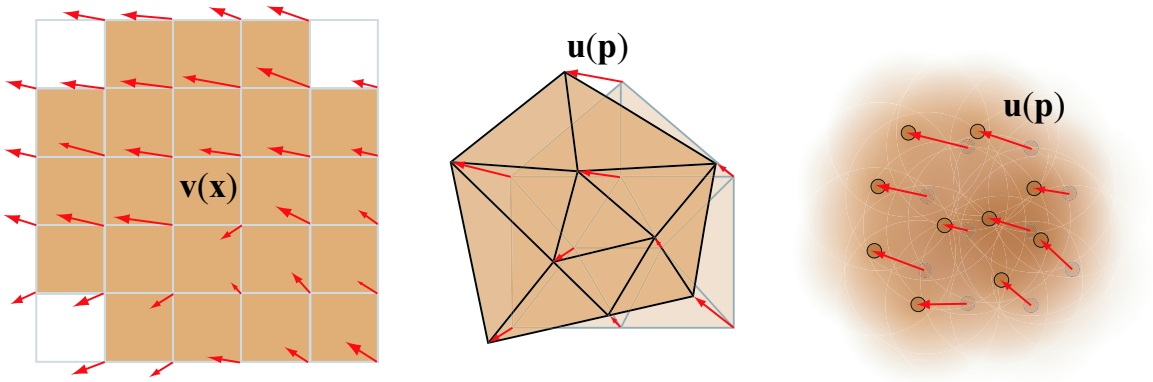


Figure 2.2: (a) Eulerian representation with velocity field $\mathbf{v}(\mathbf{x})$ at grid points $\mathbf{x}$. The shaded fields are covered by material. (b) mesh-based Lagrangian representation with displacement field $\mathbf{u}(\mathbf{p})$ at mesh nodes $\mathbf{p}$. (c) meshless Lagrangian representation with displacement field $\mathbf{u}(\mathbf{p})$ at particles $\mathbf{p}$. Each particle has a smoothing kernel assigned.

2.1.5 Viscoelastic Materials

Generally, a material is classified as solid if it resists a deformation under a weak constant stress, and as fluid if the material eventually flows. However, if temporarily high stresses are applied, a solid might be unable to resist shearing stresses anymore and therefore starts to flow (which is called *creep*), i.e., it becomes plastic. A plastic solid hence shows the typical characteristics of a fluid. On the other hand, a non-Newtonian fluid changes its viscosity depending on the applied strain rate, and thus exhibits a typical characteristic of a solid. Thus, material exists that cannot be clearly classified as either solid or fluid but can be seen as "soft solids" or "elastic liquids". A viscoelastic material is defined as a material that exhibits the characteristics of both a viscous fluid and an elastic solid. Examples are soap, honey, pudding, toothpaste, clay, and many more. The simulation of viscoelastic material is described in Chapter 6, see for instance Figure 6.13 for a comparison of different viscoelastic fluids.

2.2 Physics Representations

Many different numerical methods have been developed to solve the partial differential equations of motion described above. These approaches can be classified as Eulerian and Lagrangian methods according to the spatial or material description of the equations, respectively. Lagrangian methods can be further subdivided into mesh-based and meshless methods. In the following, the fundamental differences, advantages, and drawbacks of these approaches are discussed.

2.2.1 Eulerian versus Lagrangian Methods

Eulerian and Lagrangian methods differ in the way they look at the material. Eulerian methods evaluate the material properties at stationary points in space and compute how these properties change over time, whereas Lagrangian methods follow the moving material elements (*material coordinates*), cf. Figure 2.2. Because in Eulerian methods the mesh is fixed in space, solving the equations, for instance, using finite differences, is fast and stable. Furthermore, Eulerian methods can handle extreme deformations without changing the discretization and topological changes are captured implicitly, whereas Lagrangian methods need to adapt their discretization to avoid numerical problems. However, Lagrangian methods also have several advantages. Since the mesh is attached to the moving material, tracking is very simple and exact, and a mesh node can easily store its history. Furthermore, the object or fluid boundary is explicitly defined by the mesh, whereas with Eulerian methods the boundary has to be tracked, for instance, using level sets [OS88]. Similarly, interaction with irregular or moving boundaries or objects is often simpler with Lagrangian methods because they do not need to be discretized onto the fixed mesh as with Eulerian methods (see e.g. [CMT04]). Finally, Lagrangian methods are not restricted to a certain area in space, unlike Eulerian methods where the fixed mesh determines the simulation domain.

Due to the complementary advantages and drawbacks of Eulerian and Lagrangian methods, the method of choice depends on the simulation. Deformable objects are most often simulated using the (Lagrangian) finite element method (FEM) because boundary conditions can be solved more accurately than with Eulerian methods. On the other hand, strong deformations often yield distortions in the Lagrangian mesh, which requires complex remeshing operators to guarantee numerical robustness. Remeshing is also required if topological changes occur. Eulerian methods implicitly and stably handle strong deformations and topological changes, and therefore are standard in computer graphics for fluid simulations. However, Eulerian methods suffer from mass dissipation and alignment problems with deformable and moving boundaries. To circumvent these problems, meshless Lagrangian methods have been developed, see the next section for a discussion. Several methods exist that combine Eulerian and Lagrangian methods. For instance, the particle-in-cell (PIC) [Har63] and fluid-implicit-particle (FLIP) [BR86] method combine a Eulerian solver with particles. Other methods combine the interaction of Lagrangian deformable objects with Eulerian fluids (see e.g. [GSLF05, LIGF06]). Feldmann et al. [FOKG05] exploited the *arbitrary Lagrangian-Eulerian (ALE)* method to simulate fluids in deformable object boundaries using unstructured tetrahedral meshes [FOK05]. The ALE formulation enables the computational mesh to move with a velocity independent of the material velocity [DH04]. Recently, this scheme was extended by Klingner et al. [KFCO06] who generated the tetrahedral meshes in each time step such that they conform well to moving boundaries, therefore eliminating one of the major drawbacks of Eulerian methods. So far, this very interesting method does

not address free surfaces. Furthermore, degenerated tetrahedra can introduce instabilities similar to FEM, if they are not treated specially.

2.2.2 Mesh-based versus Meshless Methods

Mesh-based Lagrangian methods such as FEM divide a continuum into discrete elements that are connected together by a topological map, which is usually called a mesh (cf. Figure 2.2 (b)). The interpolation functions are built upon this mesh, which ensures the compatibility of the interpolation [LL02]. However, this is not always advantageous because the mesh topology is fixed and thus cannot adapt to physical changes of the continuum, especially in cases of large deformations and topology changes. Large deformations yield mesh distortions that cause severe stability and accuracy problems, for example, when using FEM [ITF04, TSIF05]. Topology changes require complex remeshing operations, which introduces numerical errors. Furthermore, maintaining a conforming mesh can be a notoriously difficult task (see e.g. [OP99]). Meshless methods remedy these problems by not storing the mesh connectivity, but instead approximate or interpolate material properties from interpolation points (also known as *collocation points*) using meshless shape functions. In meshless Lagragian methods these interpolation points, here called *particles*, move with the material (cf. Figure 2.2 (c)). As will be shown in this dissertation, the spatial discretization by volumetric particles can be efficiently and stably adapted over time according to changes in the simulation domain. The gained flexibility comes at higher computational costs for computing the meshless shape functions. Furthermore, special treatment is required for the enforcement of essential boundary conditions due to the lack of the Kronecker delta property of meshless shape functions (see [FMH04] for a survey).

2.3 Smoothed Particle Hydrodynamics

In this dissertation a meshless Lagrangian method called Smoothed Particle Hydrodynamics (SPH) is exploited. The SPH method was initially developed for the simulation of astrophysical problems such as fission of stars [GM77, Luc77]. Values of physical quantities and their spatial derivatives are approximated from neighboring interpolation points. Forces are therefore easily derived directly from the state equations. Furthermore, as a particle-based Lagrangian approach, SPH has the advantage that mass is trivially conserved and convection is dispensable. This reduces both the programming and computational complexity and is thus suitable for interactive applications.

SPH is motivated by ideas from Monte Carlo integration. Given a continuous function $A(\mathbf{x})$ defined over a domain Ω, an integral interpolant $\langle A(\mathbf{x})\rangle$ can be con-

structed using a mollification kernel ω (also called *smoothing function*)

$$\langle A(\mathbf{x})\rangle = A * \omega = \int_{\Omega} A(\mathbf{x}')\omega(\mathbf{r},h)d\mathbf{x}', \tag{2.17}$$

where the support h of the kernel indicates its smoothing scale length, $\mathbf{r} = \mathbf{x} - \mathbf{x}'$ and $d\mathbf{x}'$ is a differential volume element. Note that the interpolant reproduces $A(\mathbf{x})$ exactly if $\omega(\mathbf{r},h)$ is the delta function $\delta(\mathbf{r})$. To obtain a smooth approximation of $A(\mathbf{x})$, a kernel is used that tends to one if its support tends to zero and is normalized so that a constant function is interpolated exactly, i.e.,

$$\lim_{h\to 0} \omega(\mathbf{r},h) = \delta(\mathbf{r}), \int \omega(\mathbf{r},h)d\mathbf{x}' = 1. \tag{2.18}$$

Most often spline kernels are used that approximate the Gaussian function but have finite support. More details about the commonly used kernels and how to construct a specific kernel are given by Monaghan [Mon92] and Liu [Liu02a].

The basic idea of the SPH method is to represent a continuous field $A(\mathbf{x})$ by a Monte Carlo sampling of interacting smoothed volumetric particles. Each of the particles represents a material element of finite volume. From a mathematical point of view, particles are interpolation points from which properties of the field can be calculated. A discrete approximation of $A(\mathbf{x})$ from these particles can be achieved by replacing the differential volume element $d\mathbf{x}'$ by the volume V_j of a particle p_j with position $\mathbf{p}_j$

$$\langle A(\mathbf{x})\rangle \cong \sum_j A(\mathbf{p}_j)\omega(\mathbf{x}-\mathbf{p}_j,h)V_j \tag{2.19}$$

The discrete properties of p_j are smoothed over the finite region determined by $\mathbf{p}_j$ and V_j and hence led to the name SPH [FM03, KBLRP00]. A typical approximation of the particle volume V_i derived from Monte Carlo integration theory is

$$V_i^{-1} = \sum_j \omega(\mathbf{p}_i - \mathbf{p}_j, h). \tag{2.20}$$

If ω is a differentiable function then differentiating Equation (2.19) yields

$$\langle \nabla A(\mathbf{x})\rangle \cong \sum_j A(\mathbf{p}_j)\nabla\omega(\mathbf{x}-\mathbf{p}_j,h)V_j, \tag{2.21}$$

where $\nabla\omega = \frac{\mathbf{x}-\mathbf{p}_j}{r_j}\frac{\partial\omega}{\partial r_j}$ and $r_j = \left\|\mathbf{x}-\mathbf{p}_j\right\|$. Note that the SPH approximation of the spatial gradient of a field function is determined from the spatial derivative of the kernel function and the values at the interpolation points. Thus, the gradient of the field function does not need to be computed which is a major advantage of SPH. Different ways exist to derive the differentiable interpolant to obtain higher accuracy. For reviews on the mathematical foundation of SPH see [Ben90, Mon92, Liu02a, Mon05] and references therein.

2.3.1 SPH Approximation Error

When equations are solved numerically, the approximation $\langle A(\mathbf{x})\rangle$ of a continuous function $A(\mathbf{x})$ should be as close to A as possible. The order of an analytical solution that can be approximated without error is called *consistency order* [FM03]. To have n-th order consistency, the following conditions must be fulfilled:

$$\sum_j \omega(\mathbf{x}-\mathbf{p}_j,h)\mathbf{p}_j^m V_j = \mathbf{x}^m \quad \text{for } 0 \le m \le n, \mathbf{x} \in \Omega \tag{2.22}$$

Thus, to have 0-th order consistency, the kernel must satisfy

$$\sum_j \omega(\mathbf{x}-\mathbf{p}_j,h)V_j = 1, \mathbf{x} \in \Omega \tag{2.23}$$

This equation is fulfilled if the kernel is even and the particles are distributed equally around $\mathbf{x}$. However, during the simulation the particles become disordered. Furthermore, at the boundary of the domain, the kernel is truncated by the boundary and therefore not even anymore, even if the distribution of particles is regular. This so-called *particle deficiency problem* yields spurious boundary effects [CBJ99b]. Particle deficiency can be improved by adding ghost particles, which are created by reflecting the fluid particles with respect to the boundary [LPC+93]. Another numerical problem are *tensile instabilities* which occur when particles are under tensile stress. This results in pairwise particle clumping or even blow up of the simulation [PM85, SHA95, Bal95]. This instability can be eliminated by introducing an artificial stress force [Mon00, GMS01] or additional stress points [DRI97]. However, these approaches do not necessarily improve the numerical accuracy. Several approaches exist to improve the numerical accuracy by increasing the consistency order of the interpolation. The three most popular methods are summarized below. A nice and extensive overview of meshless methods is given by Fries and Matthies [FM03].

2.3.2 Corrective Smoothed Particle Method

To solve the particle deficiency at the boundary and reduce the tensile instability, an extension of SPH called the *Corrective Smoothed Particle Method (CSPM)* has been proposed [CBC99, CBJ99a, CBJ99b]. The idea is to normalize the function approximation such that the kernels build a partition of unity, hence the approximation is zeroth-order consistent:

$$\langle A(\mathbf{x})\rangle = \frac{1}{\sum_j \omega(\mathbf{x}-\mathbf{p}_j,h)V_j}\sum_j A(\mathbf{p}_j)\omega(\mathbf{x}-\mathbf{p}_j,h)V_j \tag{2.24}$$

This equation can be derived by looking at the Taylor series expansion for $A(\mathbf{x})$ in the vicinity of $\mathbf{p}_j$. Deriving the gradient of a field function is more involved, and makes a matrix inversion necessary [Liu02a].

2.3.3 Corrected Smoothed Particle Hydrodynamics

In the *Corrected Smoothed Particle Hydrodynamics (CSPH)* method the kernel ω is replaced by a corrected kernel $\hat{\omega}_i$ for a particle p_i

$$\hat{\omega}_i(\mathbf{x},h) = \omega(\mathbf{x},h)\alpha(\mathbf{x})[1+\beta(\mathbf{x})(\mathbf{x}-\mathbf{p}_i)], \tag{2.25}$$

where the correction parameters α and β are evaluated by enforcing the consistency conditions given in Equation (2.22) for $n = 1$ (first degree correction), see e.g. [BK00] for details.

Note that CSPM described above is a special case of CSPH. CSPM uses a constant instead of a linear correction, i.e., $\beta(\mathbf{x}) = \mathbf{0}$, yielding a corrected kernel

$$\hat{\omega}_i(\mathbf{x},h) = \frac{\omega(\mathbf{x}-\mathbf{p}_i,h)}{\sum_j V_j \omega(\mathbf{x}-\mathbf{p}_j,h)}. \tag{2.26}$$

Further correction terms for improving the pointwise integration have been proposed by Bonet and Kulasegaram [BK00].

2.3.4 Moving Least-Squares Particle Hydrodynamics

An arbitrary consistency order can be achieved using *moving least-squares (MLS)* interpolants. As described above, a smooth continuous field function $A(\mathbf{x})$ can be approximated in the form

$$\langle A(\mathbf{x})\rangle = \sum_j A(\mathbf{p}_j)\Phi_j(\mathbf{p}_j) \tag{2.27}$$

where $\Phi(\mathbf{p}_j)$ is the shape function of a particle p_j, i.e., in the SPH setting $\Phi(\mathbf{p}_j) = \omega(\mathbf{x}-\mathbf{p}_j,h)V_j$ (see Equation (2.19)). Following Fries and Matthies [FM03], $A(\mathbf{p}_i)$ can be evaluated as a Taylor series expansion

$$A(\mathbf{p}_i) = A(\mathbf{x}) + \sum_{|\alpha|=1}^{\infty} \frac{(\mathbf{p}_i-\mathbf{p}_j)^\alpha}{|\alpha|!} D^\alpha A(\mathbf{x}), \tag{2.28}$$

where $\alpha = (\alpha_1,\ldots,\alpha_d)$ is a multi-index with $\alpha_i \geq 0$ and d is the dimension of the problem domain. If α is applied to $\mathbf{x}$, then

$$\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d} \tag{2.29}$$

and $D^\alpha A(\mathbf{x})$ is the Frechet derivative of $A(\mathbf{x})$, i.e.,

$$D^\alpha A(\mathbf{x}) = \frac{\partial^{|\alpha|} A(\mathbf{x})}{\partial^{\alpha_1} x_1 \partial^{\alpha_2} x_2 \cdots \partial^{\alpha_d} x_d} \tag{2.30}$$

with $|\alpha| = \sum_{i=1}^{d} \alpha_i$ (see e.g. [RS72] for an introduction to multi-indices).

We write the shape function as a polynomial (shifted by $\mathbf{p}_i$ for computational reasons)

$$\Phi_i(\mathbf{x}) = \mathbf{b}^T(\mathbf{p}_i - \mathbf{x})\mathbf{a}(\mathbf{x})\omega(\mathbf{x} - \mathbf{p}_i, h), \tag{2.31}$$

where $\mathbf{b}(\mathbf{x}) = \{\mathbf{x}^\alpha \mid |\alpha| \leq n\}$ is a polynomial basis of consistency order n and $\mathbf{a}(\mathbf{x})$ is a vector of unknown coefficients. Plugging Equations (2.31) and (2.28) into Equation (2.27), multiplying out and rearranging leads to

$$\begin{aligned}\langle A(\mathbf{x})\rangle \;=\; & A(\mathbf{x}) \underbrace{\sum_j \mathbf{b}(\mathbf{p}_j - \mathbf{x})\mathbf{b}^T(\mathbf{p}_j - \mathbf{x})\mathbf{a}(\mathbf{x})\omega(\mathbf{x} - \mathbf{p}_j, h)}_{q_1(\mathbf{x})} + \\ & \sum_{k=2}^{n} D^{\alpha^k} A(\mathbf{x}) \underbrace{\sum_j \mathbf{b}(\mathbf{p}_i - \mathbf{x})\mathbf{b}^T(\mathbf{p}_j - \mathbf{x})\mathbf{a}(\mathbf{x}) \frac{(\mathbf{p}_j - \mathbf{x})^{\alpha^k}}{|\alpha^k|!}\omega(\mathbf{x} - \mathbf{p}_j, h)}_{q_k(\mathbf{x})} + \\ & O(\mathbf{x}^{n+1}),\end{aligned} \tag{2.32}$$

where a polynomial of degree n is used which yields an approximation order of degree n. Comparing the left and right hand side of Equation (2.32) one can see that to reach the exact solution, $q_1(\mathbf{x})$ must be equal to one while $q_k(\mathbf{x})$ must be zero for $k = 2, \ldots, n$ so that the derivatives of $A(\mathbf{x})$ cancel out. This yields a $n \times n$ system of equations for the n unknowns of $\mathbf{a}(\mathbf{x})$

$$\sum_j \mathbf{b}(\mathbf{p}_j - \mathbf{x})\mathbf{b}^T(\mathbf{p}_j - \mathbf{x})\mathbf{a}(\mathbf{x})\omega(\mathbf{x} - \mathbf{p}_j, h) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{b}(\mathbf{0}) \tag{2.33}$$

Solving this for $\mathbf{a}(\mathbf{x})$ yields the shape function of a particle p_i (after shifting the polynomial basis by adding $\mathbf{x}$)

$$\Phi_i(\mathbf{x}) = \mathbf{b}^T(\mathbf{x}) \underbrace{\left[\sum_j \mathbf{b}(\mathbf{p}_j)\mathbf{b}^T(\mathbf{p}_j)\omega(\mathbf{x} - \mathbf{p}_j, h)\right]^{-1}}_{\mathbf{M}(\mathbf{x})} \mathbf{b}(\mathbf{p}_i)\omega(\mathbf{x} - \mathbf{p}_i). \tag{2.34}$$

$\mathbf{M}(\mathbf{x})$ is called the *moment matrix*. Finally, plugging the shape function into Equation (2.27) gives

$$\langle A(\mathbf{x})\rangle = \mathbf{b}^T(\mathbf{x})\mathbf{M}^{-1}(\mathbf{x}) \sum_j \omega(\mathbf{x} - \mathbf{p}_j, h)\mathbf{b}(\mathbf{p}_j)A(\mathbf{p}_j). \tag{2.35}$$

Comparing this equation with the regular SPH Equation (2.19) shows the following relation between SPH and MLSPH

$$V_i\omega(\mathbf{x}, h) = \mathbf{b}^T\mathbf{M}^{-1}\mathbf{b}\omega(\mathbf{x}, h). \tag{2.36}$$

Therefore, $\mathbf{b}^T\mathbf{M}^{-1}\mathbf{b}$ can be interpreted as a space-dependent volume. However, this volume is purely numerical and does not have a physical or geometric meaning [Dil99b, FM03]. A careful analysis of MLSPH is given by Dilts [Dil99a].

2.4 Surface Representations

To render an animated object, a surface needs to be extracted from the simulated domain. For mesh-based Lagrangian simulation this can be simply the object's boundary. For particle-based simulations rendering the particles is usually not sufficient in computer graphics. Instead, the surface is often defined implicitly as the isosurface of a potential field from the particles [Bli82]. This surface is then either directly raytraced or converted into a triangle mesh, for instance, using the marching cubes algorithm [LC87]. For Eulerian methods, either marker particles and/or an implicit level set surface is advected with the material flow. Nowadays it is common also in Lagrangian simulations to decouple the surface representation from the volumetric representation by embedding a surface into the volume which is then advected together with the volumetric elements [MMDJ01, MG04, MBF04]. Because this is generally much faster than the actual simulation, it enables using high resolution surface for high quality rendering and a low resolution volumetric representation for fast simulation. We also make use of an embedded surface which is advected (see Section 5.5). Afterwards, the surface deforms according to implicitly and explicitly defined potential fields such that it adapts to the physics characteristics and enables user control of the surface properties (see Section 6.6).

For animating a surface one can choose between many different surface representations such as splines, polygonal meshes and level set surfaces. These have often complementary advantages and drawbacks. Following Kobbelt and Botsch [KB03], surface representations can be divided into two major classes, namely *parametric* and *volumetric* surfaces. Parametric surfaces are *explicit* representations given as the image of a parameter function, whereas volumetric surfaces are *implicitely* defined as the kernel, the so-called zero-set, of a scalar function. We will next compare implicit and explicit representations. In Section 2.5 we will discuss the basics of a point-based representation, which is used as a meshless explicit representation in this dissertation (see Sections 5.5 and 6.6). One advantage of a point-based representation is that it can be easily converted into an implicit representation based on a local projection operator (Section 2.5.1). In this dissertation we will argue that due to this efficient conversion, we are able to exploit the advantages of both implicit and explicit representations.

2.4.1 Implicit and Explicit Representations

An explicit parameterization is a mapping from a two-dimensional parameter domain Ω into 3D-space given by a function $f : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ [KB03]. It is therefore simple to generate points on the surface $\mathcal{S}$ by evaluating $f(u,v)$ for different parameter values u and v. On the other hand, it is often difficult to compute a parameterization for a given surface $\mathcal{S}$ that matches the topological and metric structure of $\mathcal{S}$.

An implicit surface is given as $\Psi = \{\mathbf{x} \in \mathbb{R}^3 \,|\, F(\mathbf{x}) = 0\}$ where $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ is a scalar function. F is usually defined such that a point $\mathbf{x}$ is inside the surface if $F(\mathbf{x}) < 0$ and outside if $F(\mathbf{x}) > 0$. Inside/outside tests thus simplify to checking the sign of $F(\mathbf{x})$. On the other hand, finding points on the surface is more advanced, which makes, for instance, rendering more difficult than with an explicit parameterization.

In computer graphics, *polygonal meshes* are the most common explicit representation, whereas *distance fields* are most often used as an implicit representation. For instance, triangle meshes are given by a set of vertices $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, a set of edges $E = \{e_1, \ldots, e_k\}$, $e_i \in V \times V$, and a set of faces $F = \{f_1, \ldots, f_m\}$, $f_i \in V \times V \times V$. Explicit representations can therefore be easily and efficiently stored in a data structure [Bau75, Kal89].

In distance fields, the function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined as the signed distance to the surface $\mathcal{S}$, hence distance computations are very simple. An implicit function is usually stored as values sampled on a 3D grid. Function values in the interior of voxels are obtained by (usually tri-linear) interpolation. Adaptive grids, which have a higher resolution close to the surface [FPRJ00], decrease the memory consumption and allow a better representation of surface features. However, since the implicit function at sharp features is not differentiable, purely implicit representations cannot represent sharp edges and corners without taking into account additional information such as the surface gradient. Furthermore, due to the missing parameterization it is very difficult to texture dynamically evolving implicit surfaces consistently.

Interesting for us are also the advantages and drawbacks of implicit and explicit representations when the surface changes dynamically. An explicit representation has the advantage that a deformation can be controlled very easily, for example, by simply changing the position of vertices. However, strong deformations yield surface distortions and therefore require remeshing. Even worse, topological changes and self-intersections have to be detected and handled explicitly, which is a difficult and time consuming problem. In contrast, because implicit surfaces are defined as a zero-set of a function, the surface is always consistent, i.e., free of geometric self-intersections, and changes its topology implicitly. Although this is often the major reason why implicit surfaces are used, it can also be a disadvantage. With implicit surfaces it is very difficult to detect and prevent unwanted topological changes, for instance, if two separated surface sheets merge because they come close.

A special class of implicit surfaces are so-called *level set surfaces* [OS88], which are state of the art in surface tracking for fluid simulation [OF02, EMF02, ELF05a]. A level set surface $\Psi(\mathbf{x},t)$ evolves always along the surface gradient (i.e., the surface normal) according to a scalar velocity function $\mathbf{v}(\mathbf{x},t)$, represented by the following partial differential equation (known as the *level set equation*)

$$\frac{\partial \Psi}{\partial t} = -\mathbf{v}|\nabla\Psi|, \tag{2.37}$$

where Ψ is often taken as the distance function, thus $|\nabla\Psi| = 1$. For level set surfaces it is sufficient to store only the function values in a narrow-band around the surface. After a deformation, the function values are updated efficiently using a fast-marching technique [Set96, Set99]. The topology can be prevented from changing by modifying the update rules [HXP01].

Implicit surfaces are usually converted into an explicit mesh using the *marching cubes (MC) algorithm* [WMW86, LC87]. A faster alternative to the volume-based MC algorithm is a surface-based approach called *marching triangles* [HSIW96], which produces higher quality triangles than MC but is not guaranteed to produce closed, manifold meshes. The *extended marching cubes (EMC)* [KBSS01] and *dual contouring* [JLSW02, SW02] algorithms enhance the MC method by using the gradient information to detect and reconstruct sharp features. An explicit representation can be converted into an implicit distance field by voxelization and computing for each voxel the signed distance to the surface [Kau87, YT02, SPG03].

Since implicit and explicit representations have distinct advantages and drawbacks, the choice depends on the application. The advantages of both can be combined by forward and backward conversion. However, with every conversion information might get lost. Thus, a better approach is to combine the two different representations and use them simultaneously. This is usually done by sampling the implicit surface with points [FGTV92]. Witkin and Heckbert [WH94] use an adaptive repulsion method to distribute the points over the surface. They employ implicit constraints to either move the points with the surface, or to move the surface with the points. Bischoff and Kobbelt [BK03a, BK03b] place sample points on the edges of the voxel grid whenever the surface is about to change its topology to guarantee topology preserving surface evolutions. To alleviate volume dissipation, Enright et al. [ELF05a] sample the narrow-band around a surface with particles. These particles are passively advected with the velocity field and guide the evolving level set surface.

In this dissertation we also propose a combination of implicit and explicit representations for surface extraction of Lagrangian animations. Our explicit representation consists of oriented surface elements, so-called *surfels* (see Section 2.5), which are directly used for rendering. This explicit representation is converted into an implicit representation using a projection operator as will be described in Section 2.5.1. Additionally, the explicit point-based surface representation is combined with an implicit representation defined by volumetric particles to exploit the

advantages of both representations, namely detailed geometry with sharp features and implicit topology changes. This approach is described in detail in Section 6.6.

2.4.2 Projecting a Point onto an Implicit Surface

As discussed above, finding points on an implicit surface is not straightforward. In this section we briefly describe the most common methods for projecting a point $\mathbf{x}$ onto an implicit surface $\Psi = \{\mathbf{x} | F(\mathbf{x}) = 0\}$, where $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ is a smooth, continuous and differentiable function. A projection operator $\psi(\mathbf{x})$ is defined as a transformation that is idempotent, i.e.,

$$\psi(\psi(\mathbf{x})) = \psi(\mathbf{x}). \tag{2.38}$$

In our case, $\psi(\mathbf{x})$ is defined such that $F(\psi(\mathbf{x})) = 0$, hence

$$\psi(\mathbf{x}) = \mathbf{x} \Leftrightarrow \mathbf{x} \in \Psi. \tag{2.39}$$

Computing $F(\psi(\mathbf{x})) = 0$ is a typical root finding problem. An excellent problem description and solutions including code can be found in [PTVF92].

The simplest case is to find the root between two given brackets, i.e., in an interval (a,b) where $F(a)$ and $F(b)$ have different signs. The *bisection method* always takes the midpoint of the interval and replaces one interval point with the midpoint depending on its sign. This guarantees to find a solution, however, convergence is only linear. Superlinear and guaranteed convergence is achieved by the *Van Wijngaarden-Dekker-Brent method* (often just called *Brent's method*), which combines root bracketing, bisection and inverse quadratic interpolation.

If only one starting point is given, one can either try to bracket the zero-level or use a method that searches the zero-level along the *steepest descent*. Neither of these methods can guarantee convergence. The most popular method of all root-finding routines is the *Newton-Raphson method* (often called *Newton's method*). This method is derived from the Taylor series expansion of a function in the neighborhood of a point. Starting with a point $\mathbf{x}$, the projection can be found by iteratively performing a Newton step

$$\mathbf{x}' \leftarrow \mathbf{x} - \nabla F(\mathbf{x}) \frac{F(\mathbf{x})}{\|\nabla F(\mathbf{x})\|}, \quad \mathbf{x} \leftarrow \mathbf{x}'. \tag{2.40}$$

This is repeated until $|F(\mathbf{x})|$ is smaller than an error threshold. If the starting point is close to the zero-level and F is sufficiently smooth, then this method converges quadratically. However, convergence is not guaranteed. Therefore, one should always check if the new point decreases $|F(\mathbf{x})|$, otherwise a smaller step along $\nabla F(\mathbf{x})$ should be taken.

A more difficult problem is to efficiently find the closest point to $\mathbf{x}$ on Ψ. This is called an *orthogonal projection*. Note that for an orthogonal projection $\psi^{\text{orth}}(\mathbf{x})$,

the gradient $\nabla F(\psi^{\text{orth}}(\mathbf{x}))$ is the normal vector of the surface Ψ at $\psi^{\text{orth}}(\mathbf{x})$ and points in direction of $\mathbf{x} - \psi^{\text{orth}}(\mathbf{x})$, i.e.,

$$\frac{\nabla F(\psi^{\text{orth}}(\mathbf{x}))}{\|\nabla F(\psi^{\text{orth}}(\mathbf{x}))\|} = \frac{\mathbf{x} - \psi^{\text{orth}}(\mathbf{x})}{\|\mathbf{x} - \psi^{\text{orth}}(\mathbf{x})\|}. \tag{2.41}$$

We can use this to iteratively solve this problem by performing a Newton step and evaluate the gradient $\nabla F(\mathbf{x}')$ at the new position $\mathbf{x}'$. A new iteration step is then performed with $\nabla F(\mathbf{x}')$ as follows:

$$\mathbf{x}' \leftarrow \mathbf{x} - \nabla F(\mathbf{x}') \frac{\|\mathbf{x} - \mathbf{x}'\| + F(\mathbf{x}')}{\|\nabla F(\mathbf{x}')\|}, \tag{2.42}$$

until the change of $\mathbf{x}'$ from one iteration to the next is smaller than an error threshold.

2.5 Point-based Representation

Point-based representations have become popular recently for shape processing [Lin01, PG01, PGK02, PKG03, WPK+04], editing [ZPKG02, AWD+04], modeling [AD03, PKKG03], rendering [ABCO+01, CH02, DVS03, ZRB+04, WTG04, BSK04, BHZK05], streaming [Paj05], 3D video [WLG04, WLW+05], and, as discussed in this dissertation, for physics-based animations [MKN+04, KMH+04, PPG04, PKA+05, KAG+05, AKP+05, WSG05]. An introduction and details to all these topics including some parts of this dissertation can be found in [GP06]. Work on point-based modeling, animation and rendering has been published recently by Adams [Ada06].

We define a point-sampled surface $\mathcal{S} = \{s_i\}_{i=1,\ldots,n}$, as a set of surface elements s_i (called *surfels*). In our case, each surfel s_i has a position $\mathbf{s}_i$, two tangent axes $\mathbf{t}^1_{s_i}$ and $\mathbf{t}^2_{s_i}$, and a set of attributes such as color. The tangent axes describe an ellipse with center $\mathbf{s}_i$ and normal $\mathbf{n}_i = (\mathbf{t}^1_{s_i} \times \mathbf{t}^2_{s_i}) / \left\|\mathbf{t}^1_{s_i} \times \mathbf{t}^2_{s_i}\right\|$.

2.5.1 Implicit Surface from Points

The point-sampled surface $\mathcal{S}$ defined above does not define a manifold due to the missing connectivity between the surfels. This is a major advantage when the surface changes dynamically because adapting the sampling of the surface is very simple and efficient. However, many surface operations require a continuously defined manifold surface. In general, it is not possible to find a *global* reference domain or parameterization for the point set. Instead, reference domains are computed *locally* and surface patches are defined over these reference domains. In his

seminal work, Levin [Lev98, Lev01, Lev04] propose a mesh-independent projection strategy based on the moving least-squares (MLS) approach. First, a local approximating tangent plane for a point $\mathbf{x}$ is computed. This frame is used as the parameter domain to fit a local polynomial to the surfels. The resulting smooth *MLS surfaces* have widely been used in computer graphics [ABCO$^+$01, ABCO$^+$03, AA03b, ZPKG02, PKKG03, FCOAS03, FCOS05]. Amenta and Kil [AK04] presented a more general definition of the MLS surface based on an energy function and a vector field, yielding so-called *extremal surfaces*.

The projection onto MLS surfaces has two disadvantages. First, computing the reference tangent plane requires solving a complicated non-linear optimization problem. Second, the projection is not orthogonal. Adamson and Alexa [AA04b] therefore proposed an implicit definition of the surface defined by the surfels, which is briefly described below.

The implicit function $F(\mathbf{x})$ of an approximating surface Ψ

$$\Psi = \{\mathbf{x} \in \mathcal{B} \,|\, F(\mathbf{x}) = 0\} \tag{2.43}$$

is defined as

$$F(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x})), \tag{2.44}$$

where $\mathbf{n}(\mathbf{x})$ is the normalized normal direction defined at $\mathbf{x}$ and $\mathbf{a}(\mathbf{x})$ is a weighted average of neighboring surfel positions $\mathbf{s}_j$

$$\mathbf{a}(\mathbf{x}) = \frac{\sum_j \omega(\left\|\mathbf{x} - \mathbf{s}_j\right\|, h)\mathbf{s}_j}{\sum_j \omega(\left\|\mathbf{x} - \mathbf{s}_j\right\|, h)}, \tag{2.45}$$

where ω is a weighting kernel with support h. The normal direction is estimated from the surfel normals

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_j \omega(\left\|\mathbf{x} - \mathbf{s}_j\right\|, h)\mathbf{n}_{s_j}}{\left\|\sum_j \omega(\left\|\mathbf{x} - \mathbf{s}_j\right\|, h)\mathbf{n}_{s_j}\right\|}. \tag{2.46}$$

Alternatively, the normal can be estimated by computing the direction of smallest weighted covariance [AA03a]. Note that Ψ is only defined in the neighborhood $\mathcal{B}$ of the surfels, where $\mathcal{B}$ is the union of a set of balls B_i with radius h centered at the surfels s_i, or more formally

$$B_i = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{s}_i\| < h\}, \tag{2.47}$$

$$\mathcal{B} = \bigcup_i B_i. \tag{2.48}$$

Gradient computation

An advantage of this implicit function definition is that the gradient of $f(\mathbf{x})$ can be computed analytically, in contrast to the MLS surface where no analytic solution

exists. Given an ortho-normal system $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2\}$, differentiating the vector fields yields the directional derivative $\nabla F(\mathbf{x}) = \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{e}_0}, \frac{\partial F(\mathbf{x})}{\partial \mathbf{e}_1}, \frac{\partial F(\mathbf{x})}{\partial \mathbf{e}_2}\right)^T$ of $F(\mathbf{x})$ with

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{e}_i} = \frac{\partial \mathbf{n}(\mathbf{x})}{\partial \mathbf{e}_i} \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x})) + \mathbf{n}(\mathbf{x}) \cdot \left(\mathbf{e}_i - \frac{\partial \mathbf{a}(\mathbf{x})}{\partial \mathbf{e}_i}\right). \tag{2.49}$$

The analytic derivation of $\frac{\partial \mathbf{n}(\mathbf{x})}{\partial \mathbf{e}_i}$ and $\frac{\partial \mathbf{a}(\mathbf{x})}{\partial \mathbf{e}_i}$ can be found in [AA04b].

Projection Operators

Given the implicit surface definition and the exact evaluation of the gradient described above, any projection method described in Section 2.4.2 can be used. When only one starting point $\mathbf{x}$ is given, the simplest projection operator is computed by projecting $\mathbf{x}$ onto the tangent plane with origin $\mathbf{a}(\mathbf{x})$ and normal $\mathbf{n}(\mathbf{x})$

$$\mathbf{x}' \leftarrow \mathbf{x} - \mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x}))\, \mathbf{n}(\mathbf{x}), \quad \mathbf{x} \leftarrow \mathbf{x}' \tag{2.50}$$

which is then repeated with the new point $\mathbf{x}'$ until $\|\mathbf{x} - \mathbf{x}'\|$ is smaller than an error threshold.

Unfortunately, this projection is not orthogonal. It can be improved by repeatedly projecting from the starting point in direction of the new normal vector $\mathbf{n}(\mathbf{x}')$

$$\mathbf{x}' \leftarrow \mathbf{x} - \mathbf{n}(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x}'))\, \mathbf{n}(\mathbf{x}'), \tag{2.51}$$

yielding a projection that is almost orthogonal since $\mathbf{n}(\mathbf{x}') \approx \nabla F(\mathbf{x}') / \|\nabla F(\mathbf{x}')\|$.

To compute an orthogonal projection, a starting point is repeatedly projected onto the tangent plane in direction of the gradient $\nabla F(\mathbf{x}')$

$$g \leftarrow \frac{\mathbf{n}(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x}'))}{\mathbf{n}(\mathbf{x}') \cdot \nabla F(\mathbf{x}')}, \quad \mathbf{x}' \leftarrow \mathbf{x} - g \nabla F(\mathbf{x}'), \tag{2.52}$$

see [AA04b] for details.

Chapter 3

State of the Art

In this chapter we summarize the state of the art in physics-based deformable modeling and fluids in computer graphics. We first give a historical background on particle systems before we describe applications of the Smoothed Particle Hydrodynamics method. Section 3.2 provides a brief overview of different mesh-based Lagrangian methods and Section 3.3 describes the state of the art in fluid simulation using Eulerian and semi-Lagrangian methods. For more details we refer to our state of the art report [NMK+06] and the discussion of relevant work given in each chapter of this dissertation. A recent survey of real-time deformable models for surgery simulation is given by Meier et al. [MLM+05]. For an exhaustive description of the state of the art in point-based graphics we refer to [GP06].

3.1 Meshless Lagrangian Methods

3.1.1 Particle Systems

Particle systems were developed by Reeves [Ree83] for explosion and subsequent expanding fire simulation in the feature film "Star Trek II: The Wrath of Khan". The same technique can also be used for modeling other fuzzy objects such as clouds and water, i.e., for objects that do not have a well-defined surface. Particles are usually graphical primitives such as points or spheres, however, they might also represent complex group dynamics such as a herd of animals [Rey87]. Each particle stores a set of properties, e.g., position, velocity, temperature, shape, age and lifetime. These attributes define the dynamical behavior of the particles over time and are subject to change due to procedural stochastic processes. Particles pass through three different phases during their lifetime: generation, dynamics and death.

In [Ree83], particles are points in 3D space that represent the volume of an object. A stochastic process generates particles in a predetermined *generation shape* that defines a region about its origin into which the new particles are randomly

placed. Property values are either fixed or may be determined stochastically. Initially, particles move outward away from the origin with a random speed. During the dynamics phase, particle properties might change as a function of time and properties of other particles. A particle's position is updated by simply adding the velocity. Finally, a particle dies if its lifetime reaches zero or if it does not contribute to the animation anymore, e.g., if it is outside of a region of interest. A particle is rendered as a point light source that adds an amount of light depending on its color and transparency property.

An advantage of particles is their simplicity, which enables the animation of a huge number of particles for complex scenes. The procedural definition of the model and its stochastic control simplifies the human design of the system. Furthermore, with particle hierarchies, complicated fuzzy objects such as clouds can be assembled and controlled. Although the particles are simulated omitting inter-particle forces, the resulting animations are convincing and fast for inelastic phenomena. The technique has thus been widely employed in movies and video games. Examples of modeling waterfalls, ship wakes, breaking waves and splashes using particle systems can be found in [FR86, Pea86, Gos90, Sim90, OH95].

Particles that interact with each other depending on their spatial relationship are referred to as *spatially coupled particle systems* [Ton92]. The interaction between particles evolves dynamically over time, thus, complex geometry and topological changes can be easily modeled with this approach. Tonnesen presented a framework for physics-based animation of solids and liquids based on dynamically coupled particles that represent the volume of an object [Ton91, Ton98]. Each particle p_i has a potential energy Θ_i that is the sum of the pairwise potential energies Θ_{ij} between p_i and all other other particles p_j, i.e.,

$$\Theta_i = \sum_{j \neq i} \Theta_{ij}. \tag{3.1}$$

The force $\mathbf{f}_i$ exerted on p_i with position $\mathbf{p}_i$ is then

$$\mathbf{f}_i = -\nabla_{\mathbf{p}_i} \Theta_i = -\sum_{j \neq i} \nabla_{\mathbf{p}_i} \Theta_{ij}. \tag{3.2}$$

So far, all particles interact with each other, resulting in $O(n^2)$ complexity where n is the number of particles. The computational costs for computing the force can be reduced to $O(n)$ when restricting the interaction to a neighborhood within a certain distance, and $O(n \log n)$ for neighbor searching (see Section 7.1). To avoid discontinuities at the neighborhood boundary, the potentials are weighted with a continuous, smooth and monotonically decreasing weighting function that depends on the distance to the particles and ranges from one at the particle position to zero at the neighborhood boundary.

For deriving inter-particle forces, the Lennard-Jones potential Θ^{LJ} is used:

$$\Theta^{\mathrm{LJ}}(r) = \frac{b}{r^n} - \frac{a}{r^m}, \tag{3.3}$$

where r is the distance between two particles, and n, m, b and a are constants. The Lennard-Jones potential is well known in molecular dynamics for modeling the interaction potential between pairs of atoms. It creates long-range attractive and short-range repulsive forces, yielding particles arranged into hexagonally ordered 2D layers in absence of external forces. A more convenient formulation, called the Lennard-Jones bi-reciprocal function, is written as

$$\Theta^{\mathrm{LJ}}(r) = \frac{-e_o}{m-n}\left(m\left(\frac{r_0}{r}\right)^n - n\left(\frac{r_0}{r}\right)^m\right), \tag{3.4}$$

where r_0 is the *equilibrium separation distance*, and $-e_0$ is the minimal potential (the magnitude is called the *dissociation energy*). Increasing the dissociation energy increases the stiffness of the model, whereas the width of the potential well is controlled with the exponents n and m. Thus, large dissociation energy and high exponents yield rigid and brittle material, whereas low dissociation energy and small exponents result in soft and elastic behavior of the object. This enables the modeling of a wide variety of physical materials ranging from stiff to fluid-like behavior. By coupling the dissociation energy with thermal energy such that the total system energy is conserved, objects can be melted and frozen. Furthermore, thermal expansion and contraction can be simulated by adapting the equilibrium separation distance r_0 to the temperature.

One problem of particle systems is that the surface is not explicitly defined. To extract a smooth surface from the particles, an algorithm is used which was developed to model electron density maps of molecular structures [Bli82]. A Gaussian potential

$$\varphi_i(\mathbf{x}) = b\mathrm{e}^{-ar^2}, \tag{3.5}$$

which is often called a *blob*, is assigned to each particle, where a and b are constants and $r = \|\mathbf{x}-\mathbf{p}_i\|$ is the distance from an arbitrary point $\mathbf{x}$ in space to the particle's position $\mathbf{p}_i$. A continuously defined potential field $\varphi(\mathbf{x})$ in space is obtained by summing the contribution from each particle

$$\varphi(\mathbf{x}) = \sum_i \varphi_i(\mathbf{x}). \tag{3.6}$$

The surface Ψ_I is then defined as an isovalue I of $\varphi(\mathbf{x})$. This yields an implicit coating of the particles, which adapts to topological changes such as splitting and merging by construction. For a more intuitive control of the surface, the constants a and b can be computed as $a = -\beta/r^2$ and $b = I\mathrm{e}^{-\beta}$, where r is the radius in isolation and β the blobiness.

The implicit coating of particles for soft inelastic substances undergoing topological changes pose challenging problems such as volume preservation, avoiding unwanted blending and contact modeling. These were addressed by Desbrun and Cani in a series of papers [Can93a, DC94, DC95]. A hybrid model is applied that is composed of two layers: Particles are used to simulate soft inelastic substances as described above, whereas an elastic implicit layer defines the surface of an object and is locally deformed during collisions. A problem of the implicit coating is

that the volume may change significantly during deformation, especially for splitting and merging. However, efficiently computing the volume of a soft object is not trivial. A *territory* of a particle p_i is defined as the (volumetric) part V_i of the object where the field contribution of p_i is the highest. Note that territories form a partition of the implicit volume of an object. Each particle samples its territory boundary by sending a fixed number of points called *seeds* in a set of distributed directions until they reach the boundary. The volume of a particle is approximated by simply summing up the distances from the particle to the seeds. The local volume variation can then be easily approximated for each particle, and the field function is changed accordingly such that the volume is preserved. Another problem is that split object parts might blend with each other when they come close. To avoid this, an *influence graph* is built at each animation step by recursively adding the neighbors of a particle that are in its sphere of influence to the same connected component. Only the particles of the same component can interact and their field functions are blended. However, the problem arises that two or more separated components might collide. For detecting a collision, the seed points on the isosurface Ψ_I are tested against the field function of another component. For resolving interpenetrations between two components with potential functions φ_1 and φ_2, exact contact surfaces are computed by applying negative compressing potentials $g_{2,1}$ and $g_{1,2}$ such that $\varphi_1 + g_{2,1} = \varphi_2 + g_{1,2} = I$, resulting in a local compression. To compensate the compression and ensure C^1 continuity of the deformed surfaces, positive dilating potentials are applied in areas defined around the interpenetration zone [Can93a, OC97]. For collision response, the compression potentials $g_{i,j}$ are computed for each colliding seed and then transmitted as response force to the corresponding particle. Additionally, the two components might be merged locally where the collision force exceeds a threshold.

Szeliski and Tonnesen introduced *oriented particles* for deformable surface modeling [ST92, Ton98], where each particle represents a small surface element (similar to *surfels*, see Section 2.5). Each particle has a local coordinate frame, given by the position of the particle, a normal vector and a local tangent plane to the surface. To arrange the particles into surface-like arrangements, interaction potentials are defined that favor locally planar or locally spherical arrangements. The Lennard-Jones potential described above is used to control the average inter-particle spacing. The weighted sum of all potentials yields the energy of a particle, where the weights control the bending and stiffness of the surface. Variation of the particle energy with respect to its position and orientation yields forces acting on the positions and torques acting on the orientations, respectively. Using these forces and torques, the Newtonian equations for linear and angular motion are solved using explicit time integration.

Recently, Bell et al. presented a method for simulating granular materials, such as sand and grains, using a particle system [BYM05]. A (non-spherical) grain is composed of several round particles, which move together as a single rigid body. Therefore, stick-slip behavior naturally occurs. Molecular dynamics (MD) is used to compute contact forces for overlapping particles. The same contact model is

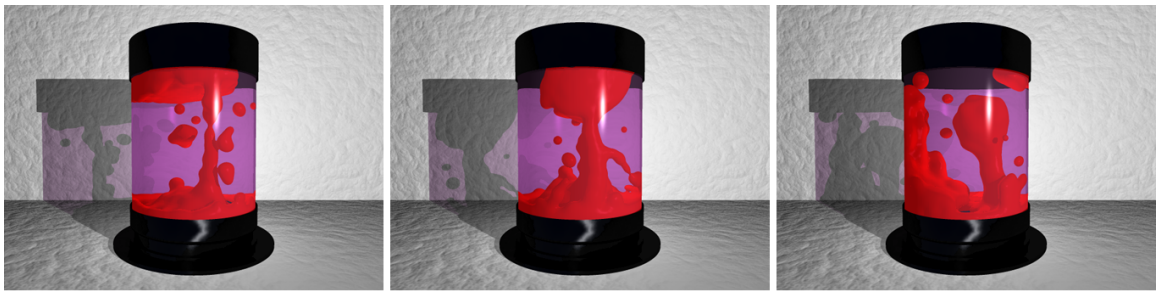


Figure 3.1: Simulation of a lava lamp using the SPH method [MSKG05].

used for collision of granular materials with rigid bodies, or even between rigid-bodies, by simply sampling the rigid body surface with particles (see Section 4.4.2 for more details).

3.1.2 Smoothed Particle Hydrodynamics

The Smoothed Particle Hydrodynamics (SPH) method is a particle-based Lagrangian technique where discrete, smoothed particles are used to compute approximate values of needed physical quantities and their spatial derivatives. Forces can be easily derived directly from the state equations. As a particle-based Lagrangian approach mass is conserved exactly and convection is dispensable. A drawback of the SPH method is that it is difficult to exactly maintain the incompressibility of material. A detailed description of the SPH method is given in Section 2.3. In the following, we will discuss applications of SPH in computer graphics.

SPH was introduced independently by Gingold and Monaghan [GM77] and Lucy [Luc77], for the simulation of astrophysical problems such as fission of stars. Stam and Fiume introduced SPH to the computer graphics community to depict fire and other gaseous phenomena [SF95]. They solve the advection-diffusion equation for densities composed of "warped blobs". Desbrun and Cani solve the state equations for the animation of highly deformable bodies using SPH [DC96]. They achieve the animation of inelastic bodies with a wide range of stiffness and viscosity. Considering the object as a set of smeared-out masses, they define the surface as an isosurface of the mass density function. To introduce surface tension and control surface characteristics such as constant volume, an active surface that evolves depending on the velocity field, similar to snakes [KWT88], is proposed [DC98]. An adaptive framework where the resolution of particles is adapted in both space and time based on a particle splitting and merging approach is presented in [DC99].

SPH has also become popular recently for fluid simulations. Stora et al. [SAC+99] animate lava flows by coupling viscosity with a temperature field and simulated heat transfer between the particles. By considering hair as a

fluid-like continuum, Hadap and Magnenat-Thalmann [HMT01] used a modified formulation of SPH to simulate hair-hair interactions. Premože et al. [PTB+03] introduced the moving particle semi-implicit method (MPS) [KTO95] to the computer graphics community for simulating incompressible multiphase fluids. Nice visual results were produced by coupling the physical particles with level sets for surface reconstruction. Müller et al. [MCG03] presented a method based on SPH and new smoothing kernels, with which fluids with free surfaces can be simulated at interactive rates with up to 5000 particles. In [MSHG04], the interaction of Lagrangian fluids and mesh-based deformable solids are modeled by placing virtual boundary particles, so-called ghost particles [Mon94], on the surface of the solid objects according to the Gaussian quadrature rule. This method is extended in [MSKG05] so that the simulation of phenomena such as boiling water, trapped air and the dynamics of a lava lamp are possible (Figure 3.1). Liquids with different polarities are simulated by computing a body force that acts perpendicular to the interface of two liquids. The force is proportional to the curvature of the interface and the surface tension. Additionally, air particles are generated and deleted dynamically where air pockets are likely to be formed, making it possible to simulate trapped air. Clavet et al. [CBP05] achieve viscoelastic fluid behavior by coupling the SPH method with springs, where plastic effects are achieved by increasing the springs' rest lengths, similar to [TPF89]. Wicke et al. [WHP+06] define the initial hexagonal lattice sampling as rest shape and compute restoring forces by matching the particle positions to their assigned lattice positions. Thus, no neighborhood information needs to be stored, making the method suitable for melting and freezing animations.

3.2 Mesh-Based Lagrangian Methods

3.2.1 The Finite Element Method

The finite element method (FEM) is a very popular and well studied method for approximating the solution of partial differential equations (PDEs), where the volume of an object is discretized using an irregular mesh (see e.g. [CMPW89] for a nice introduction). In Newtonian mechanics, the PDE has the form (see Section 2.1.1)

$$\ddot{\mathbf{x}} = f(\dot{\mathbf{x}}, \mathbf{x}, t), \tag{3.7}$$

where $\mathbf{x}(\mathbf{p},t)$ is a spatially continuous function. This function can be approximated by solving for the nodal positions $\mathbf{p}_i(t)$ of the mesh

$$\langle \mathbf{x}(\mathbf{p},t) \rangle = \sum_i \mathbf{p}_i(t)\Phi_i(\mathbf{p}), \tag{3.8}$$

where $\Phi_i(\mathbf{p})$ are *fixed* nodal basis functions that are (in the original FEM) one at node i and zero at all other nodes. Substituting $\langle \mathbf{x}(\mathbf{p},t) \rangle$ into Equation (3.7) yields

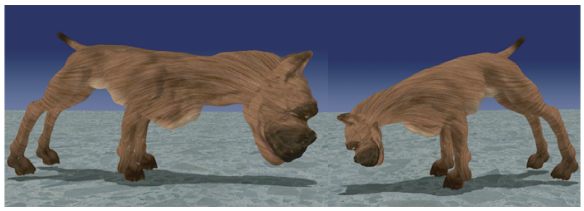


Figure 3.2: The pit bull with its inflated head (left) shows the artifact of linear FEM under large rotational deformations. The correct deformation is shown on the right [MG04]. Image courtesy of Matthias Müller, AGEIA/NovodeX.

a set of algebraic equations, which are then solved numerically. Note that choosing constant linear basis functions results in the finite difference method described in the next section.

Instead of solving this equation system, a popular method (sometimes called *explicit FEM* [DCA99, OH99]) in computer graphics is to compute nodal forces by treating the nodes like mass points in mass-spring systems, where each element is a spring connecting the adjacent nodes [OH99, DDCB01, MDM+02]. The new nodal positions are then computed by simply integrating the forces in time using an explicit or implicit scheme. For stable simulations, non-linear equations are often linearized such that an implicit integration scheme with a linear equation solver can be applied. Unfortunately, linearized elastic forces are only valid for small deformations, whereas large rotational deformations yield inaccurate restoring forces [MDM+02]. To eliminate these artifacts, Müller et al. [MDM+02, MG04] extract the rotational part of the deformation for each finite element and compute the forces with respect to the non-rotated reference frame (cf. Figure 3.2).

O'Brien et al. [OH99, OBH02b] simulated fracturing of elastic and elastoplastic material using tetrahedral finite elements with linear basis functions and explicit integration. Müller et al. [MMDJ01] achieve real-time deformations and fracture by solving for static equilibrium configurations except for collision events. Debunne et al. [DDCB01] use a linear elastic model based on the Lamé formulation for computing the deformation of an object. Applying FEM to this linear PDE results in a linear equation system. This system is solved efficiently using a hierarchy with volumetric meshes of different resolutions. Wu et al. [WDGT01] describe an adaptive nonlinear FEM simulation based on precomputed progressive meshes [Hop96]. Instead of refining the mesh elements, Grinspun et al. [GKS02] refine basis functions in their framework called CHARMS (conforming, hierarchical, adapative refinement methods). Irving et al. [ITF04] address the problem of element inversion by deriving forces from the deformation gradient. Cubical finite elements with an embedded high resolution surface have been employed by Müller et al. [MTG04] for stable and efficient fracture simulation. To support topological changes while maintaining well-shaped elements, Molino et al. [MBF04] create duplicates of the original elements which are animated as virtual nodes.

3.2.2 The Finite Difference Method

FEM approximates the solution of a differential equation, whereas the Finite Difference Method (FDM) approximates the equation itself. A major disadvantage of this simple and efficient scheme is its difficulty to approximate the boundary of an arbitrary object with a regular mesh.

Terzopoulos et al. [TPBF87] derive an energy functional as the weighted matrix norm of the difference between the metric tensors of the deformed and original. The elastic forces are computed by discretizing the continuous directional derivative of this energy term using FDM. Finally, the forces are integrated in time using semi-implicit integration. Further work covers viscoelasticity, plasticity and fracture [TF88]. Furthermore, Terzopoulos and Witkin [TW88] propose to decompose the deformation into a reference and a displacement component. The reference component is moved rigidly with the displaced component. The forces are then computed relative to this reference component to improve the numerical stability.

3.2.3 The Finite Volume Method

Although the finite volume method (FVM) is used in many CFD packages, it has been largely ignored in computer graphics. Debunne et al. [DDCB00] improved their finite difference approach for multiresolution animation of deforming objects [DDBC99] using a finite volume integration technique to approximate the Laplacian and the gradient of the divergence operators. Teran et al. [TBHF03] use FVM to simulate skeletal muscle. Similar in spirit to the geometrically motived FVM, they propose a geometric way to compute strain which leads to an intuitive way of integrating the equations of motions.

3.2.4 The Boundary Element Method

Unlike the volumetric FEM method, the Boundary Element Method (BEM) solves the equation of motion at the boundary (surface) of an object, see [Hun05] for an introduction. This is achieved by transforming the volume integral form into a surface integral by applying the Green-Gauss theorem. Thus, the 3D problem is reduced into a 2D problem, which results in a substantial speedup for solving the equation. However, BEM can only be applied to homogeneous material, and topological changes are difficult to handle.

James and Pai employ BEM for accurate real-time simulation of deformable objects. Based on a linear elastic model, reference boundary value problems are precomputed [JP99]. These are combined at run-time using a fast update method that exploits coherence. They extended this framework for the simulation of multi-zone elastokinematic models [JP02] and augment it by a multiresolution technique based on wavelets to reduce the memory requirements [JP03].

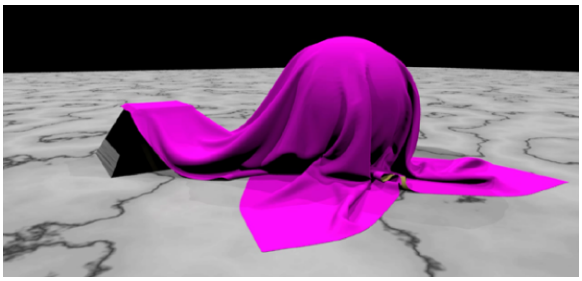


Figure 3.3: Cloth modeled using a mass-spring system [BFA02]. Image courtesy of Robert Bridson, UBC.

3.2.5 Mass-Spring Systems

Mass-spring systems discretize the space by simply connecting mass points together by a network of massless springs. This network is fixed, whereas in particle systems described in Section 3.1.1 the network is recomputed in every time step. Mass-spring systems have been popular in computer graphics for modeling faces [PB81, Wat87], soft tissues [CHP89, TW90, WT91], and the locomotion of simple creatures [Mil88, TT94]. Melting has been simulated by decreasing the spring stiffness according to a mass-points temperature, and finally the spring is removed completely [TPF89, MP89, CBP05].

Breen et al. [BHW94] state that mass-spring systems are suitable for cloth simulation due to the fact that cloth is a mechanism of warp and weft fibres, and not a continuum. Since then, mass-spring systems have dominated the cloth simulation literature [EWS96, VMT97, BW98, CK02, EGS03]. A bending model for cloth has been presented by Bridson et al. [BMF03] and for discrete shell simulation by Grinspun et al. [GHDS03]. Adaptive meshing techniques for cloth simulation have been presented in [HPH96, VB02, LV05]. Teschner et al. [THMG04] present a versatile and efficient model that can be applied for both deforming surfaces and volumes using tetrahedral and triangle meshes. Spring forces are computed from potential energy that preserve distances between vertices, the surface area of the object, and the volume of tetrahedra.

Spring constants usually need to be tuned manually because they do not directly correspond to physical values. To search for these parameters, simulated annealing [BTH+03] and a genetic algorithm [BSSH04] have been proposed.

A lot of research has been done in developing time integration scheme that preserve the large scale folding and wrinkling of cloth while stably and efficiently solving the stiff equation systems. A discussion of this topic is out of the scope of this dissertation, we refer to [NMK+06] for a nice survey.

3.3 Eulerian and Semi-Lagrangian Methods

So far, we discussed meshless and mesh-based Lagrangian methods. In this section we will look at Eulerian and semi-Lagrangian methods used for fluid simulation in computer graphics. Furthermore, we will explore how fluid boundaries are represented in the Eulerian setting. See Section 2.2 for a discussion of the fundamental differences between Eulerian and Lagrangian methods.

Foster and Metaxas popularized fluid simulation by a series of papers [FM96, FM97b,FM97a]. They solve the Navier-Stokes (NS) equations for incompressible fluids (Section 2.1.4)

$$\nabla \cdot \mathbf{v} = 0, \tag{3.9}$$

$$\frac{\partial \mathbf{v}}{\partial t} = \underbrace{\mathbf{g}}_{\text{ext. force}} \underbrace{-(\mathbf{v}\cdot\nabla)\mathbf{v}}_{\text{advection}} + \underbrace{\frac{1}{\rho}\nabla\cdot(\mu\nabla\mathbf{v})}_{\text{viscosity}} \underbrace{-\frac{1}{\rho}\nabla P}_{\text{pressure}}, \tag{3.10}$$

using finite differences on a regular voxel grid. The fluid velocity is stored on the cell faces of the grid, whereas the pressure and other scalar values are stored at the cell center to avoid pressure oscillations. This is known as *staggered* (or *MAC*) grid. For solving the NS equations, the single terms are solved separately (so-called *operator splitting*). First, the external force is simply integrated in time. Stam [Sta99] proposed a method to stably solve the (non-linear) advection term for arbitrary time steps using a semi-Lagrangian technique. The idea is to backtrack the considered grid point $\mathbf{x}^t$ in time to find the position $\mathbf{x}^{t-\Delta t}$ in the last time step $t-\Delta t$, at which a particle would have moved to $\mathbf{x}^t$. The considered attribute is then interpolated at $\mathbf{x}^{t-\Delta t}$ from the neighboring grid cell attributes at $t-\Delta t$. Finally, the attribute at $\mathbf{x}^t$ is updated with this value. Following, the (linear) viscosity term is solved, e.g., using finite differences. The obtained velocity $\tilde{\mathbf{v}}$ after these three steps is called a *best guess velocity*, where the pressure and mass conservation (Equation (3.9)) has not been taken into account yet. In the so-called *pressure projection* step, the missing pressure term is added to this velocity

$$\mathbf{v}^{\text{new}} = \tilde{\mathbf{v}} - \Delta t \nabla P, \tag{3.11}$$

where Δt is the used time step. This equation is then plugged into Equation (3.9), yielding (after rearranging the terms) a Poisson equation for the unknown pressure

$$\Delta t \nabla^2 P = \nabla \cdot \tilde{\mathbf{v}}. \tag{3.12}$$

Solving this symmetric and positive definite system of equations yields the pressure, which is then plugged back into Equation (3.11) to obtain the final $\mathbf{v}^{\text{new}}$, resulting in a divergence free (and thus incompressible) fluid. These techniques can also be applied on an adaptively refined grid, such as an octree [SY04, LGF04].

Irving et al. [IGLF06] couple a 2D technique similar to fluid methods based on height fields [KM90, OH95] with a full 3D Navier-Stokes solver at the surface. This enables speeding up the simulation of large bodies of water while capturing detailed surface motion.

To simulate smoke, Stam [Sta99] used a scalar density field to define quantities of smoke on the grid, and adds buoyancy forces based on the local smoke density. The density field is advected with the semi-Lagrangian technique described above. This approach is unconditionally stable, but the field is smoothed due to the interpolation of the values, especially on coarse grids. To reintroduce the lost small scale swirling motion in smoke simulations, Fedkiw et al. [FSJ01] add an artificial *vorticity confinement* force. A different approach was taken by Kim et al. [KLLR05] who applied *back and forth error compensation and correction* to the semi-Lagrangian technique to greatly reduce the dissipation while maintaining its stability. A different semi-Lagrangian technique for computing the advection coined *particle-in-cell* (PIC) method has been presented already in 1963 by Harlow [Har63]. The grid cells are sampled with particles. In a first step, the velocity of the particles are interpolated onto the grid cells. Second, all fluid terms except of the advection are computed on the grid. Third, the computed cell velocities are interpolated back onto the particles, which are then moved in time. Thus, the velocities are smoothed two times, yielding a very viscous fluid. Brackbill and Ruppert get rid of the smoothing in the third step by only interpolating the difference of old and new velocity of a cell back to the particles [BR86]. This approach is coined *fluid-implicit-particle* (FLIP) method. Recently, Zhu and Bridson used a model for sand in combination with either PIC or FLIP to animate sand as a fluid [ZB05], and Guendelman et al. [GSLF05] used FLIP to reduce the dissipation in the adaptive octree method [LGF04]. Selle et al. [SRF05] combine Lagrangian vortex particles with Eulerian methods to preserve small scale detail, where the vorticity confinement force is used to drive the grid based velocity field towards the particles' vorticity.

To render the fluid boundary, Foster and Metaxas [FM96] passively advect massless marker particles, which are rendered in [FM97b] as smoothed ellipsoids with an orientation based on the velocity of the particle and a normal computed from the position of the nearby particles. Carlson et al. [CMVT02] improved on this technique by splatting the particles and extract an isosurface as a polygonal mesh. For smoke visualization, Stam advects a scalar smoke density field using the semi-Lagrangian technique described above. This density field is then volume rendered. Foster and Fedkiw [FF01] were the first in computer graphics to animate the fluid boundary as level sets [OS88] (see Section 2.4.1). Level set values are only defined at Eulerian grid nodes, yielding numerical dissipation of mass in under-resolved, high curvature regions [EFFM02]. This problem has been alleviated by Enright et al. [EMF02, EFFM02, ELF05b] by sampling the fluid interface on both sides with massless particles, which are passively advected with the flow and periodically reseeded. Particles that escape the implicit surface are then used to correct errors in the level set representation. Losasso et al. [LSSF06] extend

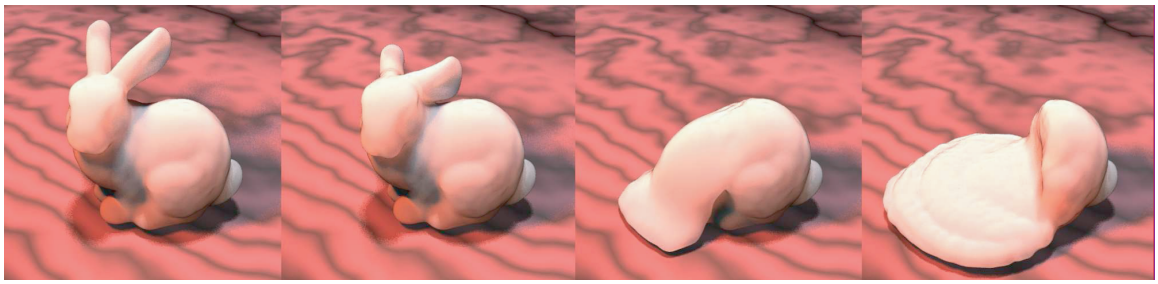


Figure 3.4: Melting bunny [CMVT02, Car04]. Image courtesy of Mark Carlson, DNA Productions, Inc.

the level set algorithm such that multiple interacting liquids can be simulated, and couple liquid and fire [NFJ02] simulation. Rasmussen et al. [REN+04] introduce two additional kind of level set particles: viscosity particles for melting, and velocity divergence particles for controlling the expansion and contraction of the liquid. Air marker particles that become trapped inside the fluid are simulated as bubbles by Greenwood and House [GH04]. Guendelman et al. [GSLF05] advect and render water marker particles that escaped the reconstructed surface as splashes and spray. Kim et al. [KCC+06] go one step further and convert these marker particles into physical particles, which are then simulated using the PIC and FLIP method described above. A description of a particle level set library including source code is given by Mokhberi and Faloutsos [MF]. An efficient scheme for storing level sets based on run-length encoding (RLE) has been proposed by Houston et al. [HWB04, HNB+05, HNB+06].

Instead of using marker particles, loss of volume can be decreased by using the error compensation and correction method described by Kim et al. [KLLR05]. Bargteil et al. [BGOS06] circumvent the dissipation problems due to interpolation of level sets by combining the advection of a distance field with an explicit surface. In every step, they compute the distance value of a grid point to the surface by backtracking the point in time using the semi-Lagrangian approach described above. The distance value of the found point is computed exactly by computing the distance of the point to the surface given as a triangle mesh. The new surface is then extracted with an adapted version of the marching cubes algorithm [LC87], where the position of the triangle vertices on the grid edges are found using a secant method. Thus, interpolation is avoided altogether.

Several extensions of the standard fluid solver have been proposed to simulate various effects. Hong and Kim [HK05b] use the ghost fluid method [Fed02] to simulate surface tension of both free and bubble surfaces by modeling the discontinuity at the interface between water and air. For obtaining accurate derivatives, both pressure and velocity are extrapolated across interfaces. Carlson et al. [CMVT02] add a temperature and variable viscosity field to simulate melting of solid objects into liquid. Solid objects are simply modeled as a fluid with very high viscosity. Solids are melted by decreasing their viscosity, which is coupled to

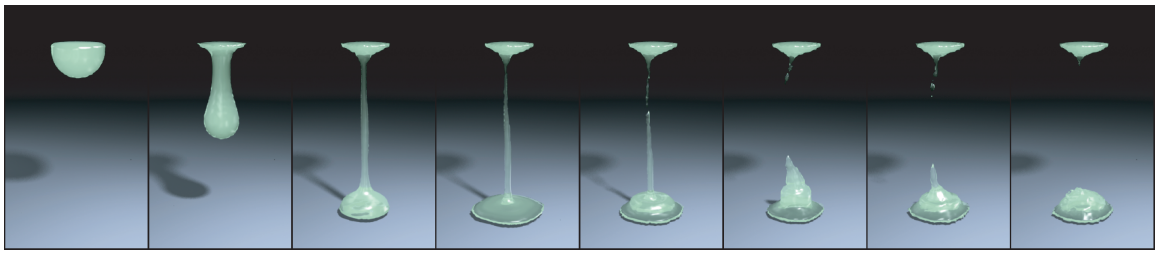


Figure 3.5: A dripping viscoelastic fluid [GBO04]. Image courtesy of Tolga G. Goktekin, UC Berkeley.

the temperature (Figure 3.4). To cope with the stiff equations due to high viscosity, an implicit integration scheme is used. Goktektin et al. [GBO04] add an elastic term to the NS equations for animating viscoelastic substances (Figure 3.5), where the strain tensor field is advected throughout the fluid grid. Carlson et al. [CMT04] simulate the interaction between rigid bodies and fluids by projecting the rigid body on a Eulerian grid and treating it as a rigid fluid. This is achieved by adding an extra term to the NS equations due to the deformation stress inside the solid. Thus, the same solver as for the fluid is used to compute the cell velocities of the rigid body, which are then used in the next time step by the rigid body solver. Discretizing the objects on the grid is avoided by Guendelman et al. [GSLF05], who couple water and smoke with thin deformable and rigid shells by casting rays from the cell center to the neighboring cell centers to find intersection with the objects. The velocity of the object at the intersection point is then used as boundary condition. The aforementioned works used a coupling technique called time splitting, where solids and fluids are computed alternately while fixing the fluid pressure and the solid's velocity, respectively. Klingner et al. [KFCO06] combine both the fluid pressure projection and the implicit solid velocity integration step into one set of equations, which are then solved simultaneously. While this increases the accuracy of the coupling and therefore allows to take larger time steps, it results in a large non-symmetric linear equation system that is composed of all fluid and solid degrees of freedom, which is computationally more expensive to solve. Losasso et al. [LIGF06] combine standard FEM solid simulation with Eulerian fluid solvers to simulate melting and burning of solids. The boundary is defined as the level set stored on the mesh nodes in material coordinates where the mesh is static. The transition from solid to fluid is achieved by adapting the nodal level set values.

The volume-of-fluid (VOF) method [HN81] stores in each voxel the volume fraction of liquid in a voxel. The interface crosses those cells that are only partially filled, i.e., where the fraction is smaller than one. This additional information can be also used to compute surface slopes and curvatures, which is needed, for instance, to simulate surface tension. Because the curvature computation is less accurate than with level sets, Sussman and Puckett [SP00, Sus03] coupled the

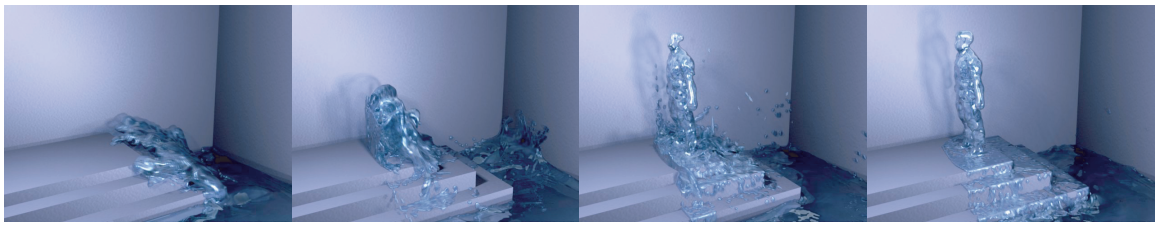


Figure 3.6: Controlled fluid simulation with detail preservation using the lattice Boltzmann method [TKPR06].

VOF and level set methods (called CLSVOF) to combine the advantages of both, namely volume preservation and smooth surfaces. Hong and Kim [HK03] model surface tension forces for rising bubbles. Takahasi et al. [TFK+03] employ the cubic interpolated propagation (CIP) [TNY85, TY87] method to simulate splashes and foam. The CIP method advects additionally to the velocity also the surface gradient to obtain a sharp interface between water and air. Song et al. [SSK05] adopt the semi-Lagrangian advection to reduce numerical dissipation, and convert dissipative cells into droplets or bubbles. Subsequently, these fragments undergo Lagrangian motion. Mihalef produce impressive results of 3D breaking waves [MMS04]. The 3D simulation is controlled using 2D slices generated by a 2D fluid solver with prescribed initial conditions derived from linear wave theory.

Instead of regular grids it is also possible to use unstructured meshes. These have the advantage that they conform better to irregular boundaries [FOK05, ETK+05]. Feldman et al. [FOKG05] used the arbitrary Lagrangian-Eulerian (ALE) method to adapt the computational mesh to deforming boundaries. Klinger et al. [KFCO06] go one step further and remesh the domain every time step such that it conforms also to moving objects.

A different approach coined lattice Boltzmann method (LBM) [FdH+87] solves the Boltzmann equation to approximate the NS equations (see [Suc01] for an overview of the method). It works similar to a cellular automaton, where the physical domain is discretized into grid cells. A cell only interacts with cells in its direct neighborhood. LBM performs well for complex geometries and, if combined with a VOF method, is fully mass conserving. Wei et al. [WZF+03, WZF+04] used LBM to simulate wind fields interacting with light-weight objects such as soap bubbles and a feather, where they exploit graphics hardware for parallel computations. Furthermore, they achieve to simulate gaseous phenomena at interactive rates [WLMK04]. Thürey and Rüde [TR04] apply LBM for interactive simulation of liquids with free surfaces. Implementation details and an adaptive time stepping scheme is given in [TKR05]. In [TKPR06], force fields defined by particles are introduced to control the fluid flow. The whole framework has been integrated into Blender [Thü06]. A hybrid simulation method that couples a 2D shallow water simulation with a 3D free surface fluid simulation, augmented with particles for the animation of drops, is given in [TRS06].

Chapter 4

Multiresolution Fluid Simulation

In this chapter a new multiresolution particle method for fluid simulation is presented. The discretization of the fluid dynamically adapts to the characteristics of the flow to resolve fine-scale visual detail, while reducing the overall complexity of the computations. We introduce the concept of virtual particles to implement efficient refinement and coarsification operators. Furthermore, a *consistent coupling* between particles at different resolution levels is achieved, i.e., our scheme guarantees that only particles of the same size interact and thus momentum is preserved. Our multiresolution method leads to speedups of up to a factor of six as compared to single resolution simulations. Our system supports multiphase effects such as bubbles and foam, as well as rigid body interactions, based on a unified particle interaction metaphor. The water-air interface is tracked with a Lagrangian level set approach using a novel Delaunay-based surface contouring method that accurately resolves fine-scale surface detail while guaranteeing preservation of fluid volume.

In our fluid model (Section 4.3), the fluid is sampled with particles which are used as interpolation points by the Smoothed Particle Hydrodynamics method (Section 2.3) for solving the Navier-Stokes equations. Additionally to water particles, air particles are generated on the fly at the water-air interface and two-way coupling between water and air is modeled by simulating cohesion at the interface (Section 4.4). With the same interaction model, also fluid-rigid body interactions can be simulated efficiently. The fluid model is enhanced with virtual particles, which provide a consistent and robust coupling between particles of different resolutions and enable to dynamically adapt the particle resolution (Section 4.5) without introducing visual discontinuities. The surface is extracted as the isovalue of a color field using Delaunay triangulation and tetrahedra marching (Section 4.6).

4.1 Introduction

Capturing the multi-scale nature of fluid flow, such as turbulence, bubbles, cohesion, and Rayleigh-Taylor instabilities, typically requires a high-resolution discretization of the computational domain. Since uniform methods, such as regular grids or constant-mass particle systems, suffer from cubic complexity and thus high computation and memory demands, spatially adaptive methods have been proposed to improve scalability. When designing such an adaptive scheme, two main questions need to be addressed: How to compute and dynamically update the adaptive discretization of the domain, and how to define the discrete differential operators on this non-uniform discretization? Clearly, an adaptive method can only be successful, if the savings in memory and processing time are not surpassed by the overhead for maintaining the adaptive spatial data structures. Similarly, an appropriate discretization of the underlying fluid flow equations is crucial to obtain a consistent and efficiently computable solution.

Losasso et al. [LGF04] recently introduced such a scheme based on an unrestricted octree decomposition. A careful design of the divergence and gradient operators yields a symmetric discretization of the Poisson equation, which can be efficiently solved even for large simulation domains (see also [SY04]). Adaptive mesh refinement [BO84, SAB+99] is a different Eulerian strategy that uses a set of uniform grids at different resolution, see [LFO05] for a comparison. Irving et al. [IGLF06] couple two and three dimensional techniques to reduce the simulation complexity. Similar in the spirit to our approach, they use a full 3D Navier-Stokes solver at the surface to capture detailed surface motion, while further away of the surface tall cells are used similar to 2D height field approaches [KM90, OH95].

Spatial adaptivity in the Eulerian setting can be problematic, however. Fluid moving through space requires constant refinement and coarsening of the underlying mesh, which leads to frequent memory updates and thus high computational cost. Consider a simple drop falling down as illustrated in Figure 4.1. Since high spatial resolution is desired close to the interface, the mesh has to be continuously refined and coarsened as the drop moves through the spatially fixed grid, even when nothing interesting is happening in terms of fluid dynamics. Another difficulty in grid-based methods is that solid boundaries have to align with the voxel structure of the grid, which can lead to aliasing artifacts for irregularly shaped domains. To address this issue, Feldman et al. [FOK05] introduced a method for simulating gases on hybrid meshes that conform to irregular domain boundaries. By mixing regular hexagonal meshes with unstructured tetrahedral meshes, their method leads to improved accuracy near irregular boundaries. Similarly, Elcott et al. [ETK+05] presented a method for simulating flow on arbitrary tetrahedral meshes using circulation preserving local operators to avoid numerical diffusion of vorticity. For simulating fluids in deforming meshes, Feldman et al. [FOKG05] use the arbitrary Lagrangian-Eulerian (ALE) formulation, which they solve using

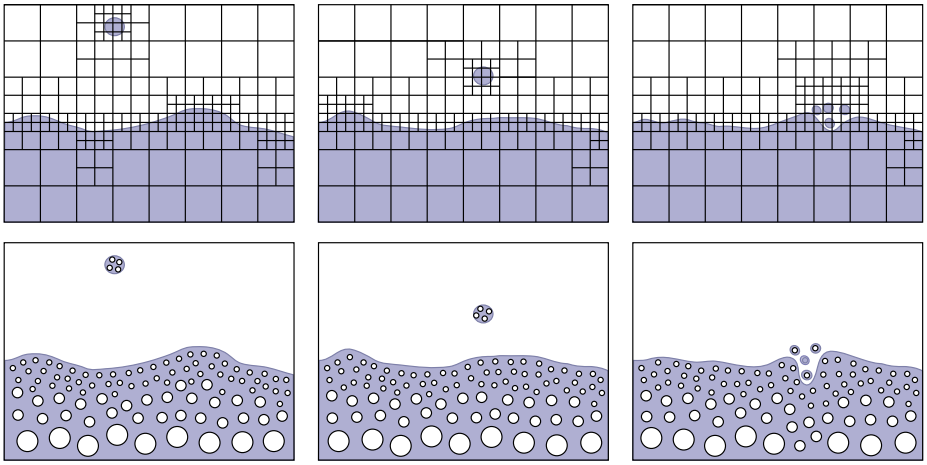


Figure 4.1: Advection of spatial detail requires substantial re-organization of adaptive Eulerian grids (top row). In Lagrangian methods updates of the discretization are directly coupled to the dynamics of the flow (bottom row).

a semi-Lagrangian method. This method has further been extended by Klingner et al. [KFCO06] who efficiently and accurately remesh the domain in every time step from scratch to conform the mesh to the boundary and moving objects. So far, this approach does not address free surfaces.

We propose a new multiresolution particle-based method that avoids grid-related aliasing artifacts as well as the complexity of maintaining conformity to dynamic object boundaries. At the same time, the spatial discretization is directly coupled to the dynamics of the flow. New particles are created near the interface to resolve visually important fine-scale surface detail. Thus, simple advection of spatial detail requires no updates in the discretization (Figure 4.1, bottom), avoiding the numerous interpolation and averaging operations that adversely affect numerical accuracy.

A crucial ingredient in our approach is a new method for coupling the interaction between particles of different size. As noted by Koumoutsakos [Kou05], variable-sized particles lead to inconsistencies in the standard mollified kernel approximation of the differential operators. We avoid this problem by allowing only a discrete set of particle sizes, analogous to an octree discretization in the Eulerian setting. The coupling between particles at different levels is achieved using a new type of *virtual* particles that guarantee the consistency of the approximation. These virtual particles at the same time provide a natural way to dynamically refine and coarsen the discretization during the simulation. Our multiresolution scheme is based on the Smoothed Particle Hydrodynamics method (see Section 2.3), but can be applied to other particle-based methods as well. We model two-phase coupling to simulate air-water interaction and extend this scheme to also implement solid-fluid and solid-solid interactions.

4.2 Related Work

State of the art Eulerian and semi-Lagrangian methods for fluid simulations are described in Section 3.3, and meshless Lagrangian methods are discussed in Section 3.1. In this section we will briefly discuss related work in surface tracking, adaptive simulations, multiphase flows, fluid-object interaction, and particle-based fluid simulations.

Most state of the art Eulerian fluid simulation algorithms use level set methods [OS88, Set99, OF03] to track the free surface. The semi-Lagrangian particle level set method [Str99, EFFM02, ELF05b] was introduced to reduce volume loss. An alternative adaptive contouring method based on octrees and backwards advection to obtain distance values was recently proposed by Bargteil et al. [BGOS06].

Adaptivity in the Eulerian setting is achieved by using an octree data structure as proposed by Losasso et al. [LGF04] and Shi and Yu [SY04]. Irving et al. [IGLF06] combine 2D and 3D solvers to speedup the simulation while having full detail close to the surface. A similar idea has been used by Thürey et al. [TRS06], who compute the full fluid flow only in a bounded region of interest that can change over time, whereas outside of this region a 2D shallow-water simulation is performed. Hong and Kim [HK03, HK05b] simulated effects such as bubbles and surface tension by modeling the discontinuity at the interface between air and water. The simulation of multiple interacting fluids is achieved in [LSSF06] by extending the particle level set algorithm. Carlson et al. [CMT04] simulate two-way coupling between rigid bodies and fluid by projecting the rigid body on the Eulerian grid. Génevaux et al. [GHD03] couple solids and fluids by computing repulsion forces between the fluid marker particles and the solid's Lagrangian nodes. Guendelman et al. [GSLF05] presented a technique for coupling water and smoke with shells. Klingner et al. [KFCO06] combine velocity and pressure constraints into one set of equations, enabling to compute the interaction between fluids and rigid bodies simultaneously, which has been extended to deformable objects by Chenatenz et al. [CGFO06]. Losasso et al. [LIGF06] simulate melting and burning of solids by coupling Eulerian fluid solvers with standard FEM solvers for deformable objects.

As an alternative to Eulerian schemes, particle-based methods for fluid simulation have recently become popular. Premoze et al. [PTB+03] used the moving-particle semi-implicit (MPS) method for solving the Navier-Stokes equations. An interactive system for water simulation using Smoothed Particle Hydrodynamics (SPH) has been presented by Müller et al. [MCG03, MSHG04, MSKG05]. Desbrun and Cani [DC96] used SPH for animating elastically and plastically deformable solids. For computational efficiency, they adapted the particle system over space and time [DC99] by enabling particles to split and merge while adapting the integration time steps automatically. Because there is no transition between refined or coarsened particles, splitting and merging introduces a discontinuity in the force computation, which is a potential source of instabilities. To speed-up

particle systems, O'Brien et al. [OFL01] propose a subdivision scheme for clustering particles and compute the dynamics on these clusters.

In particle-based methods, the free surface is most often defined as an isosurface of a color field defined by the particles [Mor00]. Hieber and Koumoutsakos [HK05a] denote this as Lagrangian particle level sets and show how both color fields and distance fields can be used to define the level set function. Müller et al. [MCG03] propose to convert SPH particles near the isovalue to surface particles for interactive visualization using surface splatting.

Recent work on SPH in computational fluid dynamics focuses on improving the accuracy by remeshing the particles onto a grid [CPK02] or regularizing the distribution of the particles [BOT01, BOT05]. Colagrossi and Landrini [CL03] show how interfacial flows such as splashing and sloshing can be accurately simulated using SPH.

4.3 Fluid Model

In our fluid framework, fluid forces are derived from the Navier-Stokes equation using SPH (see Section 2.3 for an introduction of SPH). Below we provide a brief summary and derive the relevant equations for the following sections.

The Navier-Stokes (NS) equations in the Lagrangian form are written as (see also Section 2.1.4)

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}, \tag{4.1}$$

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}. \tag{4.2}$$

The fluid density is denoted by ρ, the *pressure* P is the force per unit area that the fluid exerts on anything, μ is the (constant) *kinematic viscosity* of the fluid, $\mathbf{v}$ is the fluid velocity and $\mathbf{g}$ the acceleration due to gravity. Additional external forces (so-called *body forces*) are lumped into $\mathbf{g}$. The material derivative $\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \mathbf{v} \cdot \nabla q$ denotes the change of a physical quantity $q(\mathbf{x}, t)$ over time (see Section 2.1.1). In the Eulerian viewpoint, this is the change of q over time plus the in-flow and outflow, respectively, of this quantity at a fixed point $\mathbf{x}$ in space. In the Lagrangian viewpoint, q moves with the fluid (i.e. the particles), therefore the material derivative is equal to the partial time derivative, i.e., $\frac{Dq}{Dt} = \frac{\partial q}{\partial t}$.

The first part of the NS-equations (Equation (4.1)) is simply Newton's equation $\mathbf{f} = m\mathbf{a}$ and is therefore often called *momentum equation*. This can be easily seen by looking at the equation from a Lagrangian (particle) viewpoint. By multiplying both sides of the equation with the volume $V_i = m_i/\rho_i$ of a particle element p_i we get

$$m_i \frac{D\mathbf{v}_i}{Dt} = \underbrace{-V_i \nabla P}_{\mathbf{f}_i^{\text{pressure}}} + \underbrace{V_i \mu \nabla^2 \mathbf{v}_i}_{\mathbf{f}_i^{\text{viscosity}}} + \underbrace{m_i \mathbf{g}}_{\mathbf{f}_i^{\text{ext}}}, \tag{4.3}$$

where $\mathbf{f}_i^{\text{pressure}}$ and $\mathbf{f}_i^{\text{viscosity}}$ are the internal forces and $\mathbf{f}_i^{\text{ext}}$ the external force acting on p_i with mass m_i.

Equation (4.2) is the so-called *continuity* or *mass-conservation* equation (see Section 2.1.4). For an incompressible fluid the density does not change, i.e., $\frac{D\rho}{Dt} = 0$, and therefore the velocity field must be divergence free, i.e.,

$$\nabla \cdot \mathbf{v} = 0. \tag{4.4}$$

4.3.1 Initialization and SPH Force Approximation

The forces $\mathbf{f}_i^{\text{pressure}}$ and $\mathbf{f}_i^{\text{viscosity}}$ can be derived using the SPH approximation described in Section 2.3. Applying Equation (2.19) to the density of a particle p_i yields the so-called *summation density*

$$\langle \rho_i \rangle = \sum_j m_j \omega^{\text{poly}}(\mathbf{r}_{ij}, h), \tag{4.5}$$

where $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ is the distance vector of a particle pair. Initially, the fluid domain is sampled in a hexagonal grid which yields a closest sphere packing that is energetically ideal. All particles have the same mass m_i, which is chosen such that the average density corresponds to the physical density ρ^0 of the fluid. The volume V_i is computed as $V_i = m_i/\rho_i$.

Unfortunately, forces derived with Equation (2.21) are not symmetric. In the literature several variants of symmetrization exists, here we use the forces and smoothing kernels proposed by Müller et al. [MCG03]

$$\left\langle \mathbf{f}_i^{\text{pressure}} \right\rangle = -V_i \sum_j V_j \frac{P_i + P_j}{2} \nabla \omega^{\text{spiky}}(\mathbf{r}_{ij}, h) \text{ and} \tag{4.6}$$

$$\left\langle \mathbf{f}_i^{\text{viscosity}} \right\rangle = \mu V_i \sum_j V_j (\mathbf{v}_j - \mathbf{v}_i) \nabla^2 \omega^{\text{laplace}}(\mathbf{r}_{ij}, h). \tag{4.7}$$

The pressure P_i is computed via the constitutive equation

$$P_i = k^{\text{gas}} \left(\left(\frac{\rho_i}{\rho^0} \right)^{\gamma} - 1 \right), \tag{4.8}$$

where k^{gas} is a gas constant that determines the stiffness of the fluid and $\gamma \geq 1$ controls the relative density fluctuation. Note that these forces do not exactly enforce incompressibility, i.e., a divergence free velocity field is not guaranteed (see Equation (4.4)). The pressure force repels particles if the density ρ is larger than the rest density ρ^0, and attracts particles if $\rho \leq \rho^0$. A larger γ prevents strong density fluctuations, but requires smaller time steps. Since we operate in computer graphics with quite large time steps (Section 1.1), we choose $\gamma = 1$ (in the physics literature, usually $\gamma \approx 7$ is chosen). Other approaches approximate ρ by solving the continuity equation (4.2) directly. Whereas this so-called *continuity density*

avoids spurious boundary effects [Liu02a], it does not guarantee mass preservation exactly, and in our experiments showed to be less stable than the summation density. For the following sections we will omit $\langle\rangle$ for brevity.

4.3.2 Kernels

The kernels ω used for SPH approximations can be designed depending on the application. The standard kernel ω^{poly} used for SPH approximations in this thesis is similar to a Gaussian but has finite support h [MCG03]:

$$\omega^{\text{poly}}(\mathbf{r},h) = k^{\text{poly}} \begin{cases} \left(1-\frac{r^2}{h^2}\right)^3 & \text{if } r < h, \\ 0 & \text{otherwise,} \end{cases} \tag{4.9}$$

where $r = \|\mathbf{r}\|$ and k^{poly} is computed such that the kernel is normalized, i.e., $\int \omega^{\text{poly}}(r,h)dr = 1$. This gives $k^{\text{poly}} = \frac{315}{64\Pi h^3}$ for a kernel in 3D and $k^{\text{poly}} = \frac{4}{\Pi h^2}$ in 2D.

To avoid clustering under high pressure, Desbrun and Cani [DC96] suggested a spiky kernel with non-vanishing gradient at the center of the kernel:

$$\omega^{\text{spiky}}(\mathbf{r},h) = k^{\text{spiky}} \begin{cases} (h-r)^3 & \text{if } r < h, \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

with $k^{\text{spiky}} = \frac{15}{\Pi h^6}$ in 3D and $k^{\text{spiky}} = \frac{10}{\Pi h^5}$ in 2D.

Both of the kernels described above can have a negative second derivative. Instead of smoothing the velocity field by averaging the velocity of neighboring particles, their relative velocity might be increased as discussed by Müller et al. [MCG03]. Therefore, they propose a third kernel

$$\omega^{\text{laplace}}(\mathbf{r},h) = k^{\text{laplace}} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{r}{2h} - 1 & \text{if } r < h, \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

whose Laplacian

$$\nabla^2\omega^{\text{laplace}}(\mathbf{r},h) = k^{\text{laplace}}\left(-\frac{3r}{h^3} + \frac{2}{h^2}\right) \tag{4.12}$$

is positive everywhere and $k^{\text{laplace}} = -\frac{30}{11h^3\Pi}$ in 3D and $k^{\text{laplace}} = -\frac{30}{11h^2\Pi}$ in 2D.

4.3.3 Color Field

The boundary of the fluid discretized by particles can be found using a so-called *color field* [Mor00], which is needed for dynamically creating air particles and

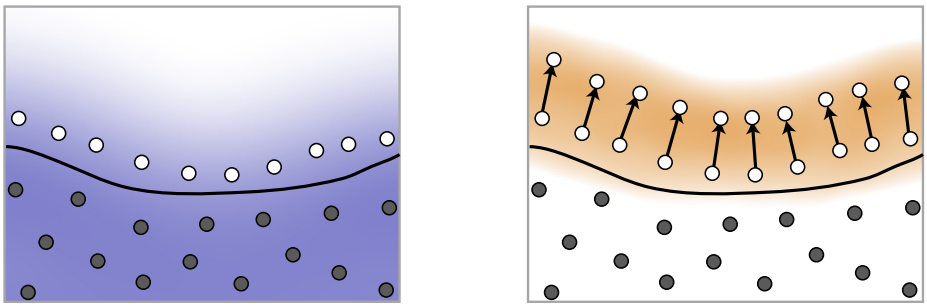


Figure 4.2: Automatic air generation. Left: color field computed by assigning the value $+1$ to water (gray) and air (white) particles. Right: color coding of magnitude of color field gradient (scaled and clamped so that white corresponds to the threshold τ_φ). The air particles near the interface generate new particles in gradient direction.

surface extraction as described below. The idea is to assign a constant value φ_i to each particle $\mathbf{p}_i$. A smooth color field $\varphi(\mathbf{x})$ is then defined using SPH as

$$\varphi(\mathbf{x}) = \sum_j V_j \varphi_j \omega^{\text{spiky}}(\mathbf{r}, h). \tag{4.13}$$

Assigning $\varphi_i = 1$ to each particle yields a color field whose gradient $\nabla\varphi(\mathbf{x})$ increases at the boundary of the particles and points in normal direction into the fluid, cf. Figure 4.2. Hence, the color field can be used for instance to determine the fluid boundary particles.

4.4 Multiphase SPH

Air influences the behavior of water both at the free surface and as air pockets (*bubbles*) enclosed by water. By simulating both fluids at the same time, two-way coupling between water and air can be achieved. The simulation of air is most important in our case at the interface, where the air is used for extracting the surface (Section 4.6) and bubbles are built. Thus, instead of having a full simulation of both fluids, the adaptivity of particle-based methods can be exploited by generating air particles dynamically during the simulation. A narrow band of air particles is maintained everywhere around the interface to model two-way coupling at the free surface. Furthermore, we extend this coupling also to solid objects, such that fluid-fluid and fluid-solid interaction are simulated using a unified particle-based approach.

4.4.1 Air Generation

For two-way coupling between water and air as well as for an accurate reconstruction of the free surface (see Section 4.6), a narrow band within a distance

$d^{\min}$ from the interface needs to be adequately sampled with air particles. Air particles further away than a distance $d^{\max}$ are deleted dynamically. Applying the color field definition described above, the color field gradient $\nabla\varphi(\mathbf{p}_i)$ is approximated using SPH for all air particles p_i with a distance smaller than $d^{\min}$. At the boundary of the narrow band, $\nabla\varphi(\mathbf{p}_i)$ is orthogonal to this boundary. A new air particle is inserted if $\|\nabla\varphi(\mathbf{p}_i)\|$ is larger than a threshold τ_φ and if the vector $\nabla\varphi(\mathbf{p}_i)$ points in opposite direction to the interface, see Figure 4.2. The position $\mathbf{p}_{\text{new}}$ of the new air particle is computed by displacing it from the position $\mathbf{p}_i$ of p_i in direction of the color field gradient:

$$\mathbf{p}_{\text{new}} = \mathbf{p}_i + s\nabla\varphi(\mathbf{p}_i)/\left\|\nabla\varphi(\mathbf{p}_i)\right\|, \tag{4.14}$$

where s is the average particle spacing, in our case $s = h/2$. For all our simulations we use $d^{\min} = h$, $d^{\max} = 2h$, and $\tau_\varphi = 0.8h$.

4.4.2 Water-Air Interaction

Between two immiscible fluids, surface tension due to molecular cohesion gives rise to a sharp pressure jump at the interface, which prevents fluids from mixing freely [HK05b]. By computing the density of the fluids using Equation (4.5) independently of each other, we get the desired discontinuity at the interface. To model the pressure jump at the interface, we add a cohesion term $-k^{\text{cohesion}}\rho_i^2$ to the pressure equation (4.8) similar to [NP00] and [CL03] and obtain

$$P_i' = k^{\text{gas}}\left(\frac{\rho_i}{\rho^0} - 1\right) - k^{\text{cohesion}}\rho_i^2. \tag{4.15}$$

The sharpness of the interface is controlled with the constant k^{cohesion}. The pressure force is then computed using Equation (4.6). When we apply this equation only for the cohesion term we get the following force

$$\mathbf{f}_i^{\text{cohesion}} = k^{\text{cohesion}} V_i \sum_j V_j \frac{\rho_i^2 + \rho_j^2}{2} \nabla\omega^{\text{spiky}}(\mathbf{r}_{ij}, h). \tag{4.16}$$

This force points in normal direction to the interface, thus giving rise to surface tension.

Rigid Bodies-Fluid Interaction

Although in this chapter we only consider water-air interaction, the method described above can be used for simulating arbitrary (immiscible) interacting fluids. In the following, we go even one step further and use the same scheme also for simulating rigid body-fluid interaction, where a rigid body is treated as a rigid

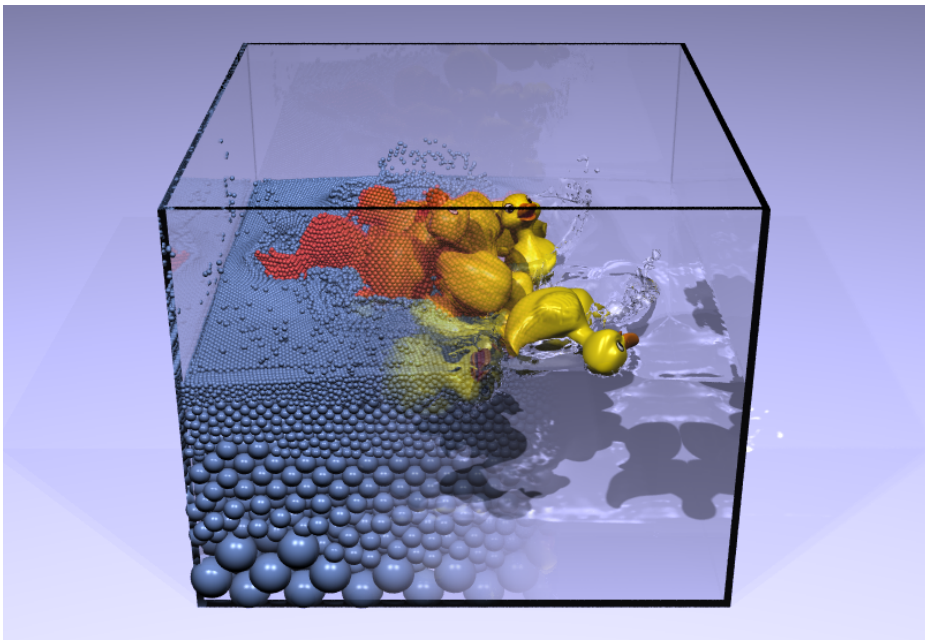


Figure 4.3: Multiresolution fluid simulation and rigid bodies-fluid interaction using a unified particle-based approach.

fluid. Thus, a rigid body is sampled with particles the same way as a fluid (see Figure 4.3), where the density and mass of the particles depend on the rigid body material properties. During the simulation, the forces acting between rigid body and fluid particles are computed exactly as for water-air interaction described above. The net force and torque acting on the rigid body are computed from the sum of its rigid body particle forces. Integration in time then yields the linear and angular velocity of the rigid body, from which we compute the new particle positions. Note that this interaction model can also be applied to deformable solids as described in the next chapter.

To simulate rigid body collisions we use the approach of Bell et al. [BYM05]. They propose to sample an offset surface of a rigid body with particles and derive the contact forces between overlapping particles from molecular dynamics. This showed to be efficient and yields very good results even for the case of rigid body stacking. Instead of sampling the surface with another set of particles, we use the rigid body volume particles for computing the contact forces.

For completeness, we give here a short description of the computed and symmetrically applied contact force between two interacting rigid body particles p_1 and p_2 of two rigid objects Γ_1 and Γ_2 (see [BYM05] for a derivation). The contact force is composed of a normal force $\mathbf{f}_{1,2}^{\text{per}}$ and shear force $\mathbf{f}_{1,2}^{\text{shear}}$. The overlap $\xi_{1,2}$ of the particles and the normalized distance vector $\hat{\mathbf{r}}_{1,2}$ is defined as

$$\xi_{1,2} = \max(0,\, 2h - \|\mathbf{p}_1 - \mathbf{p}_2\|), \tag{4.17}$$

$$\hat{\mathbf{r}}_{1,2} = (\mathbf{p}_2 - \mathbf{p}_1) / \|\mathbf{p}_2 - \mathbf{p}_1\|, \tag{4.18}$$

where h is the support radius of the particles (we use the same support as for the fluid particles). Given the relative velocity vector $\mathbf{v}_{1,2}^{\text{rel}} = \mathbf{v}_1 - \mathbf{v}_2$ of the two objects and the relative velocity in normal direction $v_{1,2}^{\text{per}} = \mathbf{v}_{1,2}^{\text{rel}} \cdot \hat{\mathbf{r}}_{1,2}$, the contact force in normal direction is computed as

$$\mathbf{f}_{1,2}^{\text{per}} = -\left(k_{1,2}^{\text{r}} \xi_{1,2}^{\frac{3}{2}} - k^{\text{d}} \xi_{1,2}^{\frac{1}{2}} v_{1,2}^{\text{per}}\right) \hat{\mathbf{r}}_{1,2}, \tag{4.19}$$

where $k_{1,2}^{\text{r}}$ is the elastic restoration coefficient controlling the stiffness and k^{d} is the viscous damping coefficient controlling the energy dissipation during collisions. Hertz theory [LL86] relates $k_{1,2}^{\text{r}}$ to the material properties as follows:

$$k_{1,2}^{\text{r}} = \frac{4}{3} E_{1,2} \sqrt{h/2} \quad \text{with} \tag{4.20}$$

$$\frac{1}{E_{1,2}} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}, \tag{4.21}$$

where E_i is Young's modulus and ν_i is Poisson's ratio of an object Γ_i (see Section 2.1.3). The shear friction force accounts for the Coulomb law with friction coefficient μ^{fri}, in combination with a viscous damping term k^{v}:

$$\mathbf{f}_{1,2}^{\text{shear}} = -\min(\mu^{\text{fri}} \mathbf{f}_{1,2}^{\text{per}},\ k^{\text{v}} \left\|\mathbf{v}_{1,2}^{\text{tan}}\right\|) \frac{\mathbf{v}_{1,2}^{\text{tan}}}{\left\|\mathbf{v}_{1,2}^{\text{tan}}\right\|}, \tag{4.22}$$

where $\mathbf{v}_{1,2}^{\text{tan}} = \mathbf{v}_{1,2}^{\text{rel}} - v_{1,2}^{\text{per}} \hat{\mathbf{r}}_{1,2}$ is the relative velocity vector in tangential direction.

4.5 Multiresolution Particle System

The multiphase SPH fluid model defined above allows complex simulations of water and its interaction with the surrounding air. The visual quality and numerical accuracy of these simulations directly depend on the number of particles, which, when sampling uniformly, increases cubically with respect to the inverse of the smallest resolved scale. Since the computational complexity is superlinear in the number of particles, high-resolution simulations quickly become intractable. Spatially adaptive sampling that concentrates more particles in regions close to the interface, can thus lead to significant savings in memory and computation time. Such a sampling scheme should also be temporally adaptive, i.e., dynamically adjust the discretization if the flow is non-stationary and detail is generated or destroyed during the simulation.

In the fluid model introduced above it is assumed that two interacting particles have the same smoothing length to guarantee *bidirectional consistency* in the particle force computation (Section 2.3). If this condition is violated, Newton's third law is no longer satisfied, i.e., particle p_i can exert a force on particle

p_j even though p_j has no influence on p_i. Thus, any spatially adaptive scheme requires an adaptation of the way particles interact to ensure conservation of momentum. A straightforward solution is to apply some averaging of the smoothing lengths or the interpolation kernels [LL03]. However, as shown by Borve et al. [BOT01], this yields errors in the order of 10% already for a factor two difference in the kernel width and leads to severe instabilities with larger kernel variations. A different approach has been coined Regularized SPH (RSPH) by Borve et al. [BOT01, BOT05]. Since SPH is derived from Monte-Carlo interpolation with the particles as interpolation points, they propose the use of additional interpolation points, either defined on a background lattice or using auxiliary particles. While substantial improvements in numerical accuracy were reported for stationary flows, dynamically changing flows are difficult to model using this method.

We propose a new, general approach for temporally and spatially adaptive particle simulations that provides a consistent approximation, while minimizing computational overhead. Similar to [BOT01], we only allow kernel widths of size $2^l h$, where $l = 0, 1, \ldots$ denotes the particle resolution level and h is the smoothing length at the highest resolution. To dynamically adapt the discretization during the simulation, particles can transition from one level to another by splitting or merging (Section 4.5.2). The key idea of our approach is the concept of *virtual* particles that serve two main purposes: They allow a consistent coupling of particles at different resolution levels and they provide a mechanism for modeling particle splitting and merging without introducing temporal visual discontinuities.

4.5.1 Virtual Particles

As will be described below, a splitting criterion ensures that neighboring particles at most differ by one level, similar to a balanced octree decomposition in the Eulerian setting. To simplify the exposition we thus only consider two resolution levels l and $l+1$, as shown in Figure 4.4 (a). Three types of *real* (i.e., non-virtual) particles are distinguished: Particles of level $l+1$ close to level l particles carry virtual level l children particles (4 in 2D, 8 in 3D). Particles of level l close to level $l+1$ particles are assigned to a virtual level $l+1$ parent particle. These two types are called *transient* particles. Other real particles of either level are *regular* particles that are not associated with any virtual particles.

SPH forces are computed only for real particles. When evaluating Equations (4.6) and (4.7), only particles of the same level, be they virtual, transient, or regular, are considered as neighbors. This means that if a particle finds a transient particle in its neighborhood, it either selects this particle or the associated virtual particle(s) when computing the interaction forces, depending on their level. If a force is computed between a real and a virtual particle, this force is symmetrically applied to the virtual particle to ensure preservation of momentum and redistributed uniformly to the associated real particle(s) (see Figure 4.4 (b)). This coupling between real and virtual particles ensures the consistency of the force

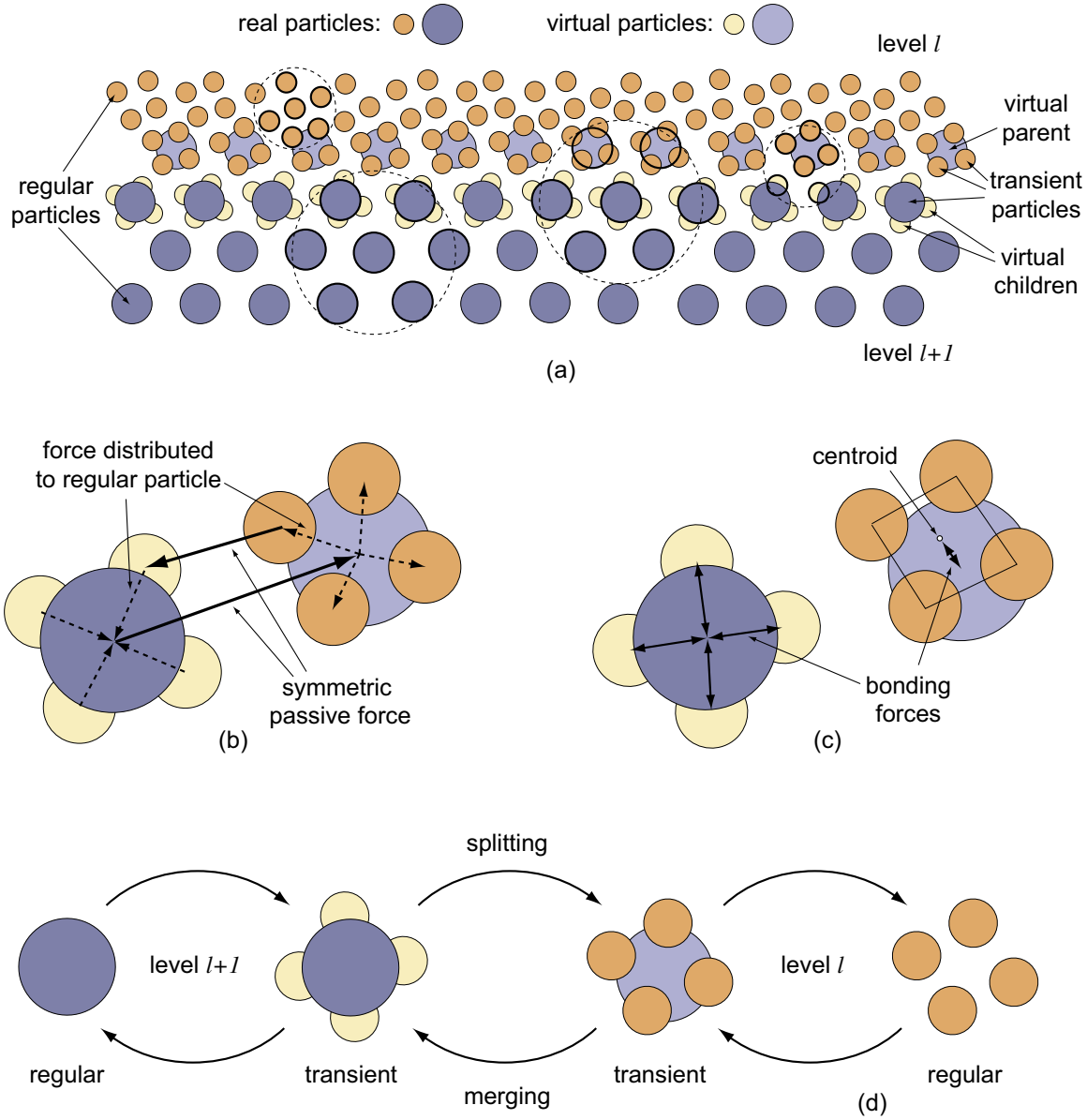


Figure 4.4: Multiresolution coupling with virtual particles. (a) two levels of particles with neighborhoods (dashed circles), (b) distribution of SPH forces, (c) bonding forces for virtual particles, (d) splitting and merging.

computations. In effect, transient particles behave like intermediates between two levels.

Virtual particles are passively advected with the flow using the SPH approximation of the velocity field. In order to keep virtual and associated real particles close together we introduce additional bonding forces that act only on virtual particles. As illustrated in Figure 4.4 (c), virtual child particles are attached to the associated transient parent particle using a linear spring force with a rest length equal to half the sample spacing of the corresponding level. Virtual parent particles are pulled toward the centroid of the associated transient child particles using a zero rest length spring.

4.5.2 Splitting and Merging

The transition of particles from one level to another is guided by two counteracting principles: On the one hand, the particle resolution should be high enough to resolve visually important fine-scale surface detail. On the other hand, we want as few particles as possible to minimize computational overhead.

The adaptation to the characteristics of the flow is achieved by enforcing that particles at the air-water interface always have smoothing length h, i.e., are at the highest resolution. This ensures that we capture all surface detail up to scale h, since our surface model is directly coupled to the particle resolution (see Section 4.6). The hierarchy is determined by ensuring that only same-sized particles interact, i.e., by guaranteeing that no real particle has regular particles of a different level within its neighborhood.

The transition from a regular level $l+1$ to a regular level l particle consists of three steps as shown in Figure 4.4 (d), from left to right: A regular level $l+1$ particle becomes a transient level $l+1$ particle, if it finds a real (regular or transient) level l particle in its neighborhood. Virtual children are created using a precomputed uniform sampling on the sphere. They are assigned half the support radius of their parent particle and 2^{-d} of its mass and volume with d the number of dimensions. If a transient $l+1$ particle has a regular level l particle in its neighborhood, its virtual children become transient real particles and itself turns into a virtual parent for its children. Transient level l particles are released from their relation with their virtual parent particle and turn into regular level l particles, if they have separated beyond their kernel smoothing length, i.e., if the maximum distance between two child particles exceeds $2^l h$. A split operation is also initiated, if all transient child particles of a common virtual parent have no virtual level l particles in their neighborhoods. Note that splitting always proceeds from coarse to fine, i.e., the splitting criterion is first evaluated for the coarse level particles. This ensures that smaller particles will not find higher level regular particles in their neighborhoods, as these would have been split already.

Merging operations proceed in the opposite direction (see Figure 4.4 (d), from right to left). In general, particles are merged whenever the new (type of) parti-

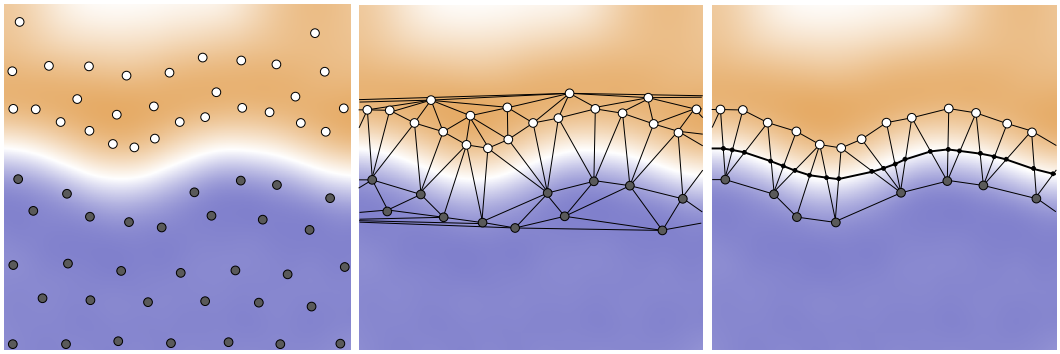


Figure 4.5: Surface extraction from the zero level set of the color field and Delaunay triangulations. Left: water particles (gray) and air particles (white) and corresponding color field. Middle: triangulation of narrow band particles. Right: surface construction from bichromatic triangles and color field.

cle potentially created by the merging operation does not fulfill the corresponding splitting criterion. The transition from regular to transient child particles requires some special treatment. To force the total number of particles to be as small as possible, even single regular particles are allowed to become transient child particles with a single virtual parent. If two such virtual parents are neighbors, their children are merged to a common parent if their total number does not exceed the allowed maximum (4 in 2D, 8 in 3D). This approach has the additional advantage that the merging can be performed incrementally and therefore a full set of eight children is not required before a merging operation can be applied. Note that this approach does not lead to undesirable fragmentation, since merging is always performed at the transition layer between two resolutions.

When a particle is split, the child particles are assigned the same velocity as the parent particle. Contrary, if a particle p_j is merged into a parent particle p_i (which has initially a velocity $\mathbf{v}_i = \mathbf{0}$), the new velocity and mass of p_i is

$$\mathbf{v}_i \leftarrow \frac{(m_i\mathbf{v}_i + m_j\mathbf{v}_j)}{m_i + m_j}, \quad m_i \leftarrow m_i + m_j. \tag{4.23}$$

This way, kinetic energy and linear momentum of the system are preserved. Angular momentum is approximately preserved because the particles are distributed about uniformly due to the pressure forces.

4.6 Surface Extraction

The fluid is sampled with water and air particles from which an interface needs to be extracted for visualization. We define the interface as the *zero level set* $\varphi(\mathbf{x}) = 0$ of the color field function $\varphi(\mathbf{x})$ described in Section 4.3.3, where we assign a constant value $\varphi_w = +1$ to water particles p_w and $\varphi_a = -1$ to air particles p_a (see also

	#real particles	#virtual particles	#air particles	#rig. body particles	spatial queries	time integration	air handling	splitting, merging	surface reconstruction	total time per frame
Sphere	170k	110k	85k	3k	64	6	9	1.1	10	91
	641k	0k			472	21		0		513
Ducks	166k	107k	73k	34k	54	9	9	1.4	8	81
	641k	0k			460	25		0		504

Table 4.1: Statistics for the animations shown in Figure 4.6 and Figure 4.7. The upper row shows the averaged data per frame for a multiresolution simulation, the equivalent single resolution results are shown in the lower row. All timings are in seconds, measured on a 2.8 GHz Pentium IV with 2 GB memory.

Figure 4.5). In Section 6.6 an algorithm is presented that constructs an explicit point-based representation from an implicit function by sampling it with surface elements (*surfels*). Here we use a different novel approach that exploits the fact that the interface separates air and water particles. The idea is to compute a *Delaunay triangulation* of the particles that have different colored particles in their neighborhood. The surface is then extracted using a variant of the marching tetrahedra algorithm [Hop94], where the surface needs to go through the bichromatic edges of two different colored particles. For a detailed description of the algorithm see [KAG+06] or [Ada06].

4.7 Results & Discussion

The crucial argument for multiresolution is scalability. While single resolution simulations in 3D require $O(m^3)$ with m the number of particles per dimension, multiresolution reduces the complexity to $O(m^2)$. Clearly, the efficiency gains depend on the characteristics of the flow. We demonstrate the effectiveness of our approach in both memory and computation time savings on two examples where a speedup of a factor of six is obtained (see also Table 4.1).

Figure 4.6 shows the break-up of a water crown and illustrates the capabilities of our multiresolution approach for resolving fine-scale detail. Preserving mass at small scales is crucial in this example as it avoids visually disturbing loss of fluid, e.g., for the drops splashing against the walls. Also note the tiny bubbles at the front of the sphere created by air particles being dragged under the water surface.

Figure 4.7 shows multiple rigid bodies, fluid, and air interacting with each other. This example demonstrates the effectiveness of our particle-based coupling method that requires only minimal adaptations to incorporate rigid or even deformable solids.

As shown in Table 4.1, the overhead for the multiresolution computation is very small (around 1%), and in fact even for simulations where all particles are on the highest level it does not increase the computational time noticeably as shown in [KAG+06]. Thus, the multiresolution approach can be applied independent of the application.

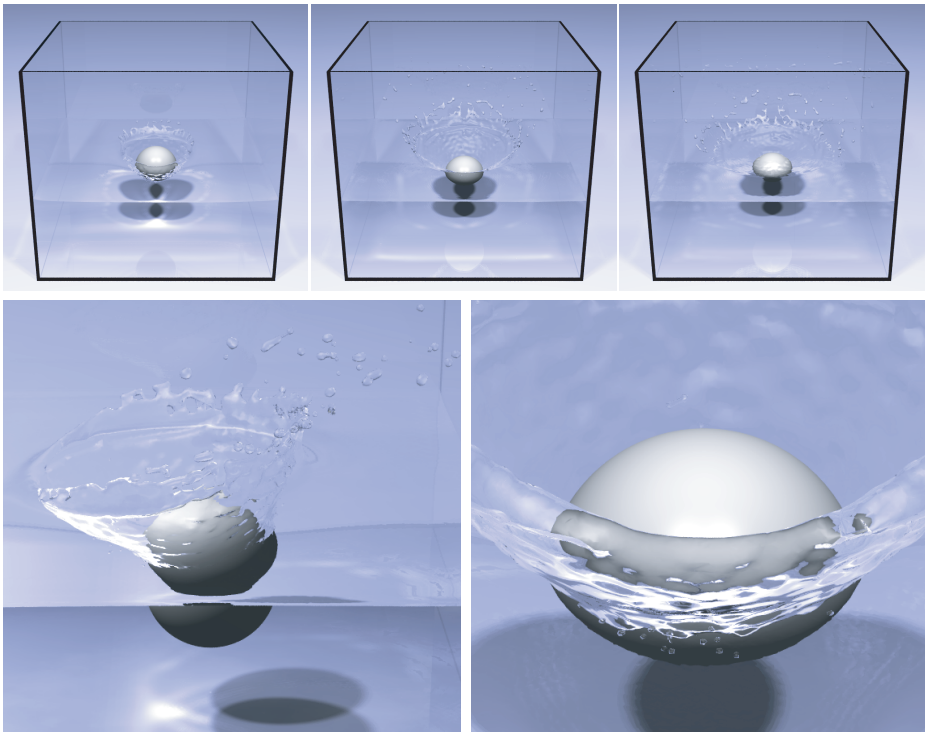


Figure 4.6: Multiphase fluid simulation of a splashing sphere. Note the small splashes (left) and bubbles (right) shown in the close-ups in the bottom row.

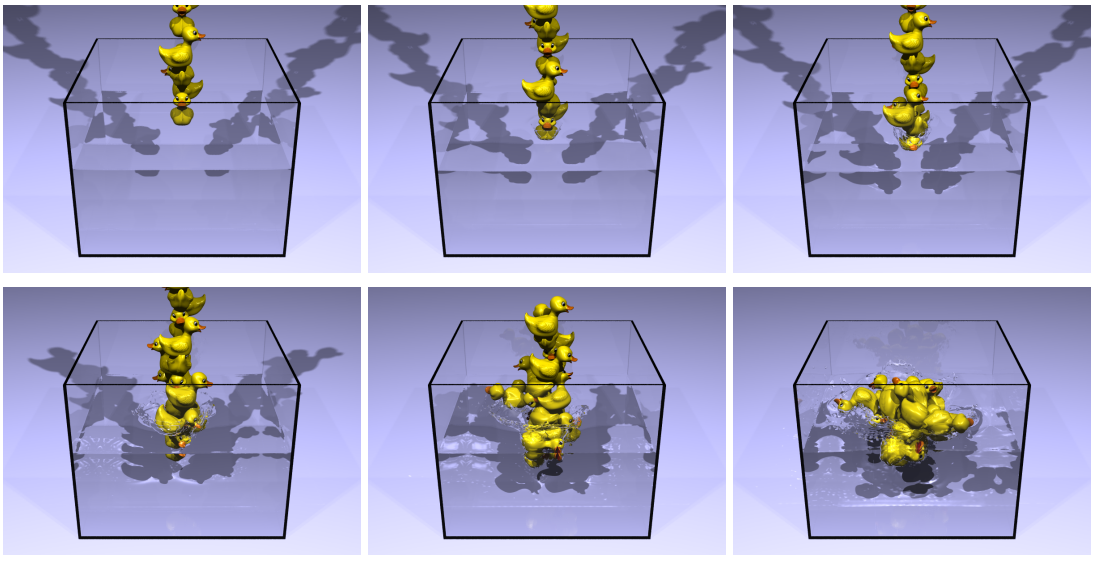


Figure 4.7: Rigid body collisions and interaction with fluid. The interaction between rigid bodies and fluids is based on a unified particle metaphor.

4.7.1 Limitations

An inherent limitation of particle-based approaches is the computational overhead caused by the particle neighborhood computations. As shown in Table 4.1, spatial queries amount to up to 80% of the total simulation time (more timings can be found in our technical report [KAG+06]). All neighborhood computations are performed using *k*d-trees as described in Section 7.1.2, which are re-initialized in each time step. Rebuilding a *k*d-tree is fast (in the range of $0.1 - 0.5$ seconds in our examples or below 0.6% of the total time), and in our experiments *k*d-trees showed to yield better performance than hash grids (Section 7.1.1). Furthermore, *k*d-trees are more suitable than hash grids in a multiresolution setting where choosing an appropriate cell size is difficult, see Section 7.1 for a thorough discussion and comparison of *k*d-trees and hash grids. We believe that substantial performance gains can be achieved using more efficient caching schemes that exploit temporal coherence. Nevertheless, the implicit neighborhood relations of regular grids are clearly a performance advantage of Eulerian methods. Another strength of grid-based approaches is incompressibility. While methods for SPH have been proposed to make the velocity field (approximately) divergence free [CR99], the computational overhead is substantial. However, the choice of the stiffness constant k^{gas} (we use $k^{\text{gas}} = 400\text{k}$) in Equation (4.8) is non-trivial as it depends on the time step (1ms in our simulations). With larger k^{gas}, the fluid is less compressible but the time step needs to be reduced to guarantee a stable simulation. Another limitation is that due to the smoothing inherent in the SPH method, fine-scale turbulences are often damped out.

4.8 Extensions & Future Work

Our multiresolution fluid framework can be extended in various ways.

A view-dependent fluid simulation could potentially lead to additional performance gains. Additional to the splitting criteria described in Section 4.5.2, the distance to the camera and the view frustum could be used as a splitting and merging criteria to reduce the resolution in regions that are not visible or far away from a certain viewpoint.

Another possible application of our multiresolution method is for hybrid molecular-continuum simulations (see e.g. [Kou05] for an overview). The algorithms would need to be adapted such that virtual particles can differ more than a factor of two to their associated real particles.

The presented multiresolution approach proved to be stable and yields plausible results in our empirical experiments. However, to make this approach useful also for physically accurate simulations as in CFD, an analytical error analysis and qualitative comparisons between multi- and single-resolution animation are required.

In Section 4.4 we discussed interaction of liquid and air. Similarly, liquid-liquid

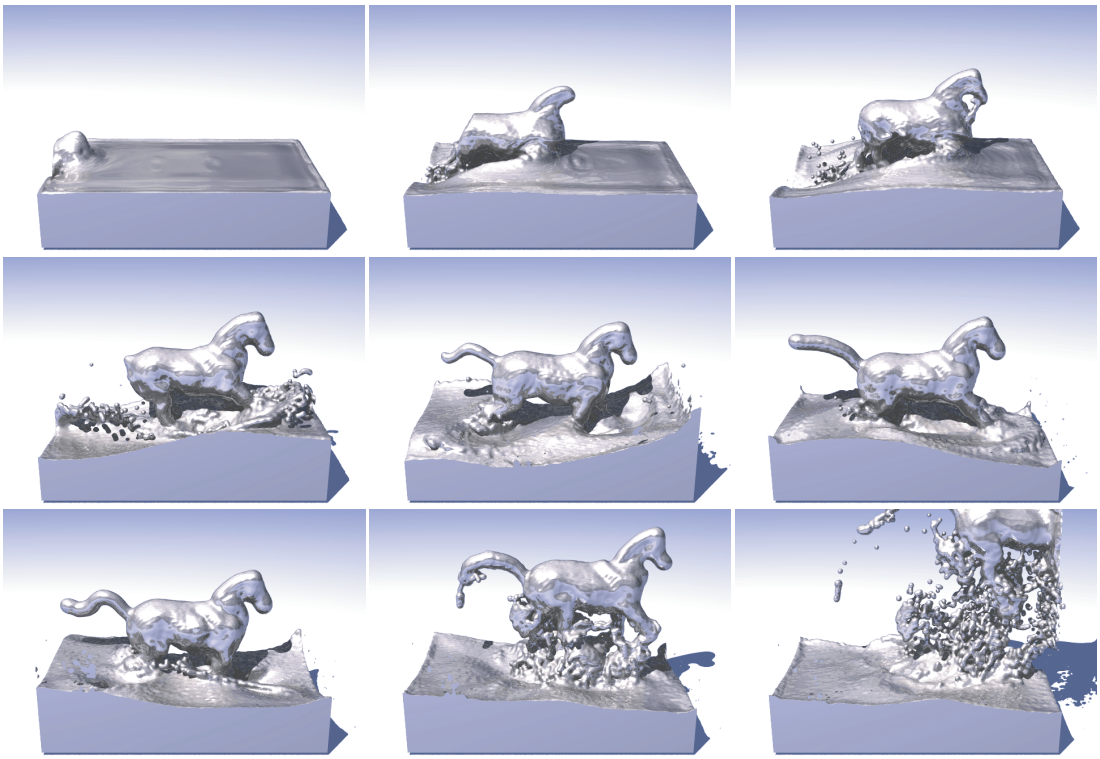


Figure 4.8: Fluid following the shape of an animated model using the SPH method. Control particles are created by sampling the interior of the horse.

interactions can be simulated. In [MSKG05], we propose methods to simulate the interaction of multiple fluids with different characteristics, such as immiscible liquids. Furthermore, dynamic phase changes are modeled by simply changing the attributes of particles. To avoid a full SPH simulation of air as described in this chapter, we generate air only where air pockets are likely to be formed. These techniques enable the simulation of phenomena such as boiling water, trapped air and the dynamics of a lava lamp (cf. Figure 3.1).

Our interface tracking method allows the advection of surface parameter information, which could be used to advect surface textures. Given a bichromatic edge, we can compute the intersection of this edge with the surface of the previous time step and transfer the parameter coordinates to the current time step, similar to the approach of Bargteil et al. [BGOS06]. This can be done efficiently by exploiting that most bichromatic edges in one time step also exist in the next and therefore only a small amount of edges needs to be tested for intersection.

In Chapter 6 we will show how the framework can be extended to enable two-way coupling between deformable objects and fluids, and simulate melting of deformable objects to fluids and solidifying fluids to solid objects. This poses great challenges for the surface extraction.

In [TKPR06], we propose a new fluid control technique that uses scale-dependent force control to preserve small-scale fluid detail. Control particles

define local force fields and can be generated automatically from either a physical simulation (Figure 3.6) or a sequence of target shapes (Figure 4.8). We use a multi-scale decomposition of the velocity field and apply control forces only to the coarse-scale components of the flow. Small-scale detail is thus preserved in a natural way avoiding the artificial viscosity often introduced by force-based control methods.

4.9 Summary

Methods have been presented for the animation of multiresolution particle-based fluids. Immiscible fluids are prevented from mixing by simulating the pressure jump at the interface, rising surface tension. This enables two-way coupling between water and air to simulate multiphase effects such as bubbles and foam. The same model is used for simulating the interaction between fluids and solids, where the solid's volume is also sampled with particles. Based on molecular dynamics forces, these solid particles are also used for rigid body collision handling. Thus, the simulation of fluids and rigid bodies is based on a unified particle metaphor, which will be extended in the next chapter to handle also deformable solids.

With the proposed multiresolution method, the discretization of the fluid volume dynamically adapts to the characteristics of the flow to resolve fine-scale visual detail. The spatially and temporally adaptive sampling leads to substantial savings in memory and computation time, thus enabling significantly more complex flow simulations with the same processing resources. The concept of virtual particles achieves a consistent coupling of particles at different levels and provides an efficient mechanism for dynamic resampling. It is lightweight, simple to implement, and can easily be incorporated into other particle-based simulation environments. In the next chapter, the multiresolution approach will be exploited for dynamically adapting the resolution to robustly handle strong volume deformations and fracturing of elastic objects.

Using a meshless Lagrangian simulation technique revealed several advantages compared to Eulerian methods. Mass is tracked exactly with the particles, thus preventing mass dissipation. Air particles are created dynamically where needed, and particles in air pockets are simulated as bubbles. Updates of the discretization are efficient and directly coupled to the dynamics of the simulation. Similarly, our novel Delaunay-based interface tracking method enables advection of the surface together with the particles and efficiently generates high-quality surfaces that automatically adapt to the resolution of the simulation. Thus, even tiniest bubbles and splashes are captured correctly. Drawbacks of the SPH method are the approximate incompressibility and difficulties in handling thin sheets of liquid. Furthermore, neighborhood computations require a significant amount of the whole computation time.

Chapter 5

Deformable Solid Simulation

In this chapter a method is presented for modeling and animating a wide spectrum of elastic objects with material properties anywhere in the range form stiff elastic to highly plastic, including brittle and ductile fracture as well as contact handling. In our system, both the volume and the surface representation are point-based. Central to our method is a highly dynamic surface and volume sampling, which enables large deviations of an object from the original shape, supports arbitrary crack initiation, propagation and termination, and the computation of a consistent contact surface for collision response. The point-based surface representation is decoupled from the physics representation, therefore a coarse volumetric representation can be used in combination with a highly detailed object surface, which enables efficient, stable and high quality animations.

In our elasticity model (Section 5.4), which is derived from continuum mechanics (Section 5.3), the spatial derivatives of the displacement field and elastic body forces are computed from the local neighborhood of the volumetric particles. The surface of the object is sampled with oriented surface elements (hereafter called *surfels*), which are advected along with the particles (Section 5.5). This framework is extended to incorporate fracturing by creating complex fracture patterns of interacting and branching cracks according to the internal stress, where the propagating crack fronts directly affect the coupling between particles (Section 5.6). To guarantee numerical stability also for strong deformations and frequent fracturing, the sampling of the simulation domain is adapted on the fly (Section 5.7). For handling contact between deformable objects, collision handling and deformation are decoupled, yielding stable and efficient collision response (Section 5.8). Putting all parts together and exploiting dynamic caching, an efficient and stable framework is obtained for the simulation of elasto-plastic solids including fracturing and collision (Section 5.9). Finally, extensions of this framework and possible directions for future research are presented (Section 5.10).

5.1 Introduction

The animation of deformable objects in virtual systems has been a challenging problem for years and combines several research subjects in computer graphics. For instance, in a virtual surgery simulator, human body parts are simulated as deformable bodies (so-called *soft objects*) based on measurements and experiments, where the materials span a wide range from stiff elastic to highly plastic. A surgeon might interact with the body parts using a haptics device. These then deform due to the external forces exerted from the haptics device and might collide with other objects. Alternatively, the surgeon might cut parts of with a virtual blade. Due to the impact of the blade or colliding objects, parts might get damaged and fracture. As this simple example shows a framework for deformable solid simulation must be able to handle elasto-plastic deformations of objects due to external forces and collisions, fracturing due to high internal stresses (cutting is a special case of fracturing where the blade is considered to be a user manipulated crack), and contact handling between colliding objects. Other important requirements on a virtual system are robustness and interactivity, i.e., the methods have to be both stable and efficient.

A majority of previous simulation methods in computer graphics use 2D and 3D meshes. Most of these approaches are based on mass-spring systems, or the more mathematically motivated finite element (FEM), finite difference (FDM) or finite volume (FVM) methods, in conjunction with elasticity theory. In mesh-based approaches, complex physical effects, such as fracturing, melting, and solidifying, pose great challenges in terms of restructuring. Additionally, under large deformations the original meshes may become arbitrarily ill-conditioned. For the simulation of these complex physical phenomena, efficient and consistent surface and volume representations are needed that allow simple restructuring. Meshless Lagrangian methods have several advantages over finite element methods. Most importantly, meshless methods avoid complex remeshing operations and the associated problems of element cutting and mesh alignment sensitivity common in FEM. Maintaining a conforming mesh can be a notoriously difficult task when the topology of the simulation domain changes frequently [OP99]. Repeated remeshing operations can adversely affect the stability and accuracy of the calculations, imposing undesirable restrictions on the time step. Finally, meshless methods are well suited for handling large deformations due to their flexibility when locally refining the sampling resolution. Our goal is to unify the simulation of materials ranging from stiff elastic to highly plastic into one framework, using a meshless, point-based volume and surface representation which entirely omits explicit connectivity information and thus implicitly encompasses the complex physical effects described above.

5.2 Related Work

In this section we give a short overview of existing work on physics-based deformable models that is most relevant for us, including fracturing and collision handling.

5.2.1 Deformable Modeling

Pioneering work in the field of physics-based animation was carried out by Terzopoulos and his co-workers [TPBF87, TF88, TW88, TPF89]. They compute the dynamics of deformable models from the potential energy stored in the elastically deformed body using finite difference discretization.

A large number of *mesh-based* methods for both off-line and interactive simulation of deformable objects have been proposed in the field of computer graphics. Examples are mass-spring systems used for cloth simulation [BW98, DSB99], the boundary element method (BEM) [JP99] and the finite element method (FEM), which has been employed for the simulation of elastic objects [DDCB01, GKS02], plastic deformation [OBH02b] and fracture [OH99].

Desbrun and Cani were among the first to use *meshless* models in computer graphics. In [DC95], soft, inelastic substances that can split and merge are animated by combining particle systems with simple inter-particle forces and implicit surfaces for collision detection and rendering. The Smoothed Particle Hydrodynamics (SPH) method (see Section 2.3) is applied in [DC96]: discrete particles are used to compute approximate values of physical quantities and their spatial derivatives. Space-adaptivity is added in [DC99]. Szeliski and Tonnesen [ST92, Ton92, Ton98] derived elastic inter-particle forces using the Lennard-Jones potential energy function (commonly used to model the interaction potential between pairs of atoms in molecular dynamics) for surface modeling and physics-based animation of deformable solids.

5.2.2 Fracturing

Terzopoulos et al. extended their work on deforming objects using finite differences [TPBF87] to handle plastic materials and fracture effects [TF88]. Mass-spring models [HTK98] and constraint-based methods [SWB00] have also been popular for modeling fracture in graphics, as they allow for easy control of fracture patterns and relatively simple and fast implementations.

Recent efforts have focused on finite element methods that directly approximate the equations of continuum mechanics. O'Brien et al. were the first to apply this technique for graphical animation in their paper on brittle fracture [OH99]. Using element cutting and dynamic remeshing, they adapt the simulation domain to conform with the fracture lines that are derived from the principal stresses. [OBH02a]

introduces strain state variables to model plastic deformations and ductile fracture effects.

Element splitting has also been used in virtual surgery simulation, where Bielser et al. [BGTG03] introduced a state machine to model all configurations of how a tetrahedron can be split. Müller et al. [MMDJ01, MG04] demonstrate real-time fracturing using an embedded boundary surface to reduce the complexity of the finite element mesh. The virtual node algorithm of Molino et al. [MBF04] combines the ideas of embedding the surface and remeshing the domain. Elements are duplicated and fracture surfaces are embedded in the copied tetrahedra. This allows more flexible fracture paths, but avoids the complexity of full remeshing and associated time stepping restrictions.

5.2.3 Contact Handling

Collision detection for deformable surfaces has recently gained increasing attention. A survey of recent research is presented by Teschner et al. [TKH+05].

Many different approaches exist for collision handling of deforming objects. Penalty methods, pioneered by Moore and Wilhelms [MW88], are probably the most widely used solutions in computer graphics. However, these methods cannot ensure that the objects do not penetrate. Baraff and Witkin [BW92] use a constraint-based method to prevent objects from penetration. This requires solving a linear complementary problem (LCP), which is computationally expensive for complex objects. Impulse-based methods [Hah88, MC95] assume short contacts only, and therefore are not suitable for soft objects. Desbrun and Cani [Can93b, DC95] presented a system for animation and collision handling of implicit surfaces generated by skeletons. Exact contact surfaces are achieved by deforming the implicit layer. The compression of the surface yields response forces which are transmitted to the skeleton. However, the generated implicit models tend to be blobby.

Point-sampled object surfaces have become popular in recent years. In this context, the problem of collision detection and response has only been addressed very recently. Collision detection of point-sampled objects was first handled in our shape modeling system [PKKG03, Kei03]. Klein and Zachmann [KZ04] presented an approach for time-critical collision detection of point clouds using a sphere bounding hierarchy. However, they do not deal with collision response. Pauly et al. [PPG04] model contact for point-sampled quasi-rigid objects, i.e., rigid objects with an elastic layer at the surface. They compute exact contact surfaces by setting up linear complementarity constraints and solving for the tractions that act on these surfaces. The wrench on the rigid object is then computed from these tractions.

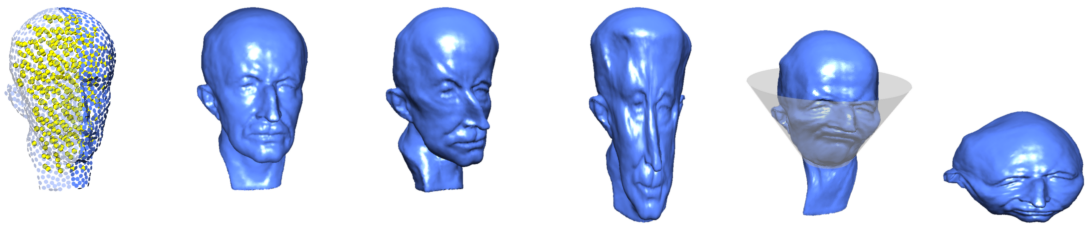


Figure 5.1: The physical volume elements (particles in yellow) and the surface elements (surfels in blue) are both represented as points. The model presented in the Sections 5.3–5.5 allows the simulation of elastic, plastic, melting, and solidifying objects (from left to right).

5.3 Continuum Mechanics Equations

The continuum elasticity equations describe how to compute the elastic stresses inside a volumetric object, based on a given deformation field [Coo95, Chu96]. Consider a model of a three-dimensional body whose material coordinates are $\mathbf{m} = (x, y, z)^T$. To describe the deformed body in world coordinates, a continuous displacement vector field $\mathbf{u}(\mathbf{m}) = (u(\mathbf{m}), v(\mathbf{m}), w(\mathbf{m}))^T$ is used where the scalar displacements $u(\mathbf{m})$, $v(\mathbf{m})$ and $w(\mathbf{m})$ are functions of the material coordinates. The world coordinates $\mathbf{x}$ of a point with material coordinates $\mathbf{m}$ are $\mathbf{x} = \mathbf{m} + \mathbf{u}(\mathbf{m})$ in the deformed model. The Jacobian of this mapping is given by

$$\mathbf{J} = \mathbf{I} + \nabla \mathbf{u}^T = \begin{bmatrix} u_{,x} + 1 & u_{,y} & u_{,z} \\ v_{,x} & v_{,y} + 1 & v_{,z} \\ w_{,x} & w_{,y} & w_{,z} + 1 \end{bmatrix}, \tag{5.1}$$

with the following column and row vectors

$$\mathbf{J} = \left[\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z\right] = \begin{bmatrix} \mathbf{J}_u^T \\ \mathbf{J}_v^T \\ \mathbf{J}_w^T \end{bmatrix}. \tag{5.2}$$

The subscripts with commas represent partial derivatives (e.g. $u_{,x} = \frac{\partial u(\mathbf{m})}{\partial x}$). To measure strain, we use the quadratic Green-Saint-Venant strain tensor

$$\varepsilon^{\mathrm{s}} = \mathbf{J}^T \mathbf{J} - \mathbf{I} = \nabla \mathbf{u} + \nabla \mathbf{u}^T + \nabla \mathbf{u} \nabla \mathbf{u}^T. \tag{5.3}$$

This symmetric quadratic tensor has the advantage that it is rotation invariant, in contrast to the linearized version most often used in computer graphics for performance reasons, which fails for larger deformations (see e.g. [MDM+02]). In our model we assume a Hookean material, i.e., a linear relationship between force and deformation (see Section 2.1.3):

$$\sigma^{\mathrm{s}} = \mathbf{C}\, \varepsilon^{\mathrm{s}}, \tag{5.4}$$

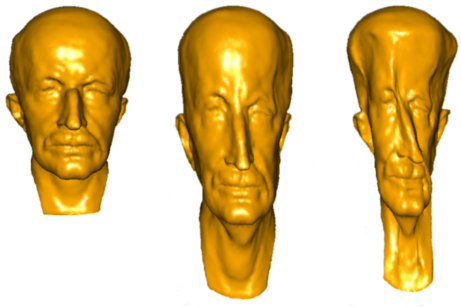


Figure 5.2: The effect of Poisson's ratio: the undeformed model (left) is stretched using a Poisson ratio of zero (middle) and 0.49 (right).

where $\mathbf{C}$ is a rank four tensor, approximating the constitutive law of the material, and both ε^s and σ^s are symmetric 3×3 (rank two) tensors. For an isotropic material, $\mathbf{C}$ has only two independent coefficients, namely Young's modulus E and Poisson's ratio ν (Section 2.1.3). Therefore, Hooke's law is written as

$$\begin{bmatrix} \sigma^s_{xx} \\ \sigma^s_{yy} \\ \sigma^s_{zz} \\ \sigma^s_{xy} \\ \sigma^s_{yz} \\ \sigma^s_{zx} \end{bmatrix} = k^s \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{bmatrix} \begin{bmatrix} \varepsilon^s_{xx} \\ \varepsilon^s_{yy} \\ \varepsilon^s_{zz} \\ \varepsilon^s_{xy} \\ \varepsilon^s_{yz} \\ \varepsilon^s_{zx} \end{bmatrix} \tag{5.5}$$

with $k^s = \frac{E}{(1+\nu)(1-2\nu)}$. Anisotropic materials can be simulated by modifying $\mathbf{C}$ accordingly.

The elastic body forces can be computed via the strain energy density $\tilde{U}^{\text{strain}}$ (energy per unit volume), which is the potential energy stored in a deformed material

$$\tilde{U}^{\text{strain}} = \frac{1}{2}(\varepsilon^s \cdot \sigma^s) = \frac{1}{2}\left(\sum_{i=1}^{3}\sum_{j=1}^{3} \varepsilon^s_{ij}\sigma^s_{ij}\right). \tag{5.6}$$

The elastic force per unit volume at a point $\mathbf{m}$ is the negative gradient of the strain energy density with respect to this point's displacement $\mathbf{u}(\mathbf{m})$ (the *directional derivative* $\nabla_{\mathbf{u}}$). For a Hookean material, this expression is written as

$$\tilde{\mathbf{f}}^{\text{elastic}} = -\nabla_{\mathbf{u}}\tilde{U}^{\text{strain}} = -\frac{1}{2}\nabla_{\mathbf{u}}(\varepsilon^s \cdot \mathbf{C}\,\varepsilon^s) = -\sigma^s \nabla_{\mathbf{u}}\varepsilon^s. \tag{5.7}$$

5.4 Elasticity Model

In order to use the continuous elasticity equations described above in a numerical simulation of a dynamic elastic object, we need to discretize the volume of the

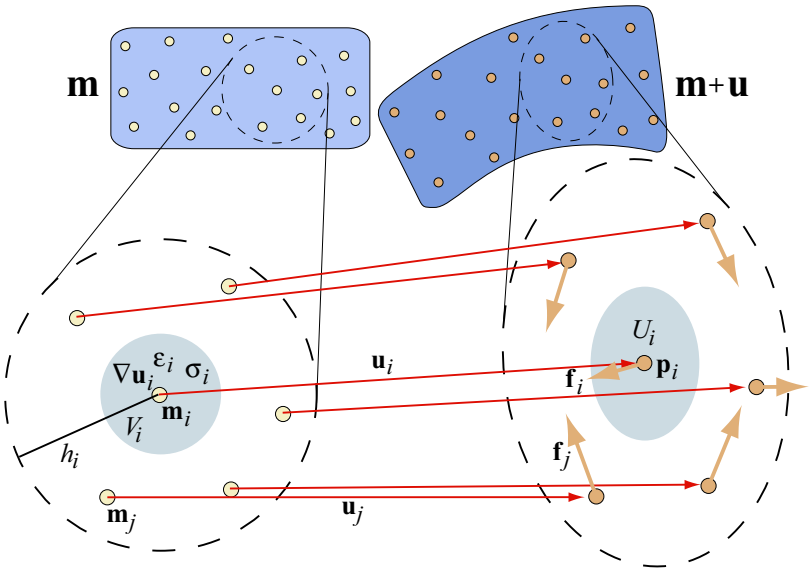


Figure 5.3: As a basic unit, we consider a particle with material coordinates $\mathbf{m}_i$ and its neighbors at $\mathbf{m}_j$ within distance h_i. The gradient of the displacement field $\nabla \mathbf{u}$ is computed from the displacement vectors $\mathbf{u}_i$ and $\mathbf{u}_j$, the strain ε_i^s from $\nabla \mathbf{u}_j$, the stress σ_i^s from ε_i^s, the strain energy U_i^{strain} from ε_i^s, σ_i^s and the volume V_i and the elastic forces as the negative gradient of U_i^{strain} with respect to the displacement vectors.

object into volumetric elements dV. Similarly to the fluid model described in Section 4.3, the object is sampled with smoothed particles p_i with position $\mathbf{m}_i$ in material coordinates and fixed mass m_i. The sampling algorithm is described in Section 5.7. The density ρ_i is approximated using SPH according to Equation (4.5) and the particle volume is computed as $V_i = m_i/\rho_i$, see Section 4.3.1 for details.

In our meshless framework, quantities of a particle p_i, such as strain ε_i^s, stress σ_i^s, strain energy U_i^{strain} and the elastic body force $\mathbf{f}_i^{\text{elastic}}$, are computed from its neighbors p_j which are placed within its support radius h_i (cf. Figure 5.3). While the stress is in our case a linear function of the strain, the strain is computed from the spatial derivative of the displacement, i.e., $\nabla \mathbf{u}_i$. We will next discuss how we compute $\nabla \mathbf{u}_i$ using a first order approximation scheme, and then show how the strain energy and elastic forces are computed.

5.4.1 Moving Least-Squares Approximation of $\nabla \mathbf{u}$

For computing ε_i^s, the spatial derivative of the displacement field $\nabla \mathbf{u}_i$ at $\mathbf{m}_i$ is needed. To guarantee zero elastic forces for rigid body modes, the approximation of $\nabla \mathbf{u}_i$ from the displacement vectors $\mathbf{u}_j$ of the neighboring particles must be at least first order accurate. Hence, the moving least-squares (MLS) method with a

linear basis can be used, which yields first order accurate interpolation of point-sampled functions, see also Section 2.3.4. In the following, we will only consider the x-component u of the displacement field $\mathbf{u}(\mathbf{m}) = (u(\mathbf{m}), v(\mathbf{m}), w(\mathbf{m}))^T$ and $\mathbf{m} = (x, y, z)^T$. The basic idea is to approximate a continuous scalar field $\mathbf{u}(\mathbf{m})$ in the neighborhood of $\mathbf{m}_i$ using a Taylor approximation:

$$u(\mathbf{m}_i + \Delta\mathbf{m}) = u_i + \nabla u|_{\mathbf{m}_i} \cdot \Delta\mathbf{m} + O(\|\Delta\mathbf{m}\|^2), \tag{5.8}$$

where $\nabla u|_{\mathbf{m}_i} = (u_{,x}, u_{,y}, u_{,z})$ at particle p_i with material coordinates $\mathbf{m}_i$ (the index after the coma denotes a spatial derivative). For particles p_j close to p_i we get a first order approximation $\langle u_j \rangle$ of the values u_j as

$$\langle u_j \rangle = u_i + \nabla u|_{\mathbf{m}_i} \cdot \mathbf{r}_{ij} = u_i + \mathbf{r}_{ij}^T \nabla u|_{\mathbf{m}_i}, \tag{5.9}$$

where $\mathbf{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})^T = \mathbf{m}_j - \mathbf{m}_i$. The weighted least-squares error e_i of the approximation $\langle u_j \rangle$ is given by

$$e_i = \sum_j (\langle u_j \rangle - u_j)^2 \omega_{ij}, \tag{5.10}$$

where $\omega_{ij} = \omega(\mathbf{r}_{ij}, h_i)$ is a smoothing kernel (we use the spiky kernel ω^{spiky}, see Equation (4.10)). Substituting Equation (5.9) into Equation (5.10) and expanding yields

$$e_i = \sum_j (u_i + u_{,x}\, x_{ij} + u_{,y}\, y_{ij} + u_{,z}\, z_{ij} - u_j)^2\, \omega_{ij}. \tag{5.11}$$

We want to find the unknowns $u_{,x}$, $u_{,y}$ and $u_{,z}$ such that the error e_i is minimized. Therefore, the derivatives of e_i with respect to $u_{,x}$, $u_{,y}$ and $u_{,z}$ are set to zero, yielding three equations for the three unknowns

$$0 = 2 \sum_j \mathbf{r}_{ij} \left(u_i + \mathbf{r}_{ij}^T \nabla u|_{\mathbf{m}_i} - u_j \right) \omega_{ij}. \tag{5.12}$$

Multiplying out and rewriting yields

$$\left(\sum_j \mathbf{r}_{ij}\, \mathbf{r}_{ij}^T \omega_{ij} \right) \nabla u|_{\mathbf{m}_i} = \sum_j (u_j - u_i) \mathbf{r}_{ij} \omega_{ij}. \tag{5.13}$$

Finally, the spatial derivatives of $u(\mathbf{m})$ at $\mathbf{m}_i$ are obtained as

$$\nabla u|_{\mathbf{m}_i} = \mathbf{M}_i^{-1} \left(\sum_j (u_j - u_i) \mathbf{r}_{ij} \omega_{ij} \right), \tag{5.14}$$

where the inverse of the so-called *moment matrix* $\mathbf{M}_i = \sum_j \mathbf{r}_{ij} \mathbf{r}_{ij}^T \omega_{ij}$ is used for computing the derivatives of v and w as well. Note that $\mathbf{M}_i$ of a particle p_i needs to be recomputed only if the neighborhood of p_i changes due to resampling (Section 5.7), introduced discontinuities (Section 5.6.2), or if the material position of the particles changes due to a (non-linear) update of the rest shape (Section 6.4.1).

5.4.2 Elastic Force Computation

Given the spatial derivative $\nabla \mathbf{u}_i$ of a particle p_i we can update the Jacobian $\mathbf{J}_i$, the strain ε_i^s and the stress σ_i at the particle position $\mathbf{m}_i$ using Equations (5.1), (5.3) and (5.4):

$$\mathbf{J}_i \leftarrow \begin{bmatrix} \nabla u|_{\mathbf{m}_i}^T \\ \nabla v|_{\mathbf{m}_i}^T \\ \nabla w|_{\mathbf{m}_i}^T \end{bmatrix} + \mathbf{I}, \quad \varepsilon_i^s \leftarrow (\mathbf{J}_i^T \mathbf{J}_i - \mathbf{I}), \quad \sigma_i^s \leftarrow (\mathbf{C}\, \varepsilon_i^s). \tag{5.15}$$

Based on Equation (5.6) the strain energy stored around a particle p_i is estimated as

$$U_i^{\text{strain}} = \frac{1}{2} V_i (\varepsilon_i^s \cdot \sigma_i^s) \tag{5.16}$$

assuming that strain and stress are constant within the rest volume V_i of particle p_i, equivalent to using linear shape functions in FEM. The strain energy is a function of the displacement vector $\mathbf{u}_i$ of p_i and the displacements $\mathbf{u}_j$ of all its neighbors. Taking the derivative with respect to these displacements using Equation (5.7) yields the force $\mathbf{f}_{ii}^{\text{elastic}}$ acting at particle p_i and the forces $\mathbf{f}_{ij}^{\text{elastic}}$ acting on all its neighbors p_j

$$\mathbf{f}_{ii}^{\text{elastic}} = -\nabla_{\mathbf{u}_i} U_i^{\text{strain}} = -V_i \sigma_i^s \nabla_{\mathbf{u}_i} \varepsilon_i^s, \tag{5.17}$$

$$\mathbf{f}_{ij}^{\text{elastic}} = -\nabla_{\mathbf{u}_j} U_i^{\text{strain}} = -V_i \sigma_i^s \nabla_{\mathbf{u}_j} \varepsilon_i^s \tag{5.18}$$

The force $\mathbf{f}_{ii}^{\text{elastic}}$ acting on p_i turns out to be the negative sum of all $\mathbf{f}_{ij}^{\text{elastic}}$ acting on its neighbors p_j, i.e., $\mathbf{f}_{ii}^{\text{elastic}} = -\sum_j \mathbf{f}_{ij}^{\text{elastic}}$. Thus, the total force

$$\mathbf{f}_i^{\text{elastic}} = \mathbf{f}_{ii}^{\text{elastic}} + \sum_k \mathbf{f}_{ki}^{\text{elastic}} \tag{5.19}$$

acting on particle p_i conserves linear and angular momentum. Note that the sum is not over the neighbors of p_i but over all particles p_k that have p_i in their support radius h_k. In our case where all particles have the same support radius h, the particles p_k are exactly the neighbors of p_i and

$$\mathbf{f}_i^{\text{elastic}} = 2 \sum_j \mathbf{f}_{ij}^{\text{elastic}}. \tag{5.20}$$

Applying Equation (5.14), Equations (5.17) and (5.18) can be further simplified to the compact form (see [MKN+04] for details)

$$\mathbf{f}_{ii}^{\text{elastic}} = \mathbf{F}_e \mathbf{M}_i^{-1} \left(-\sum_j \mathbf{r}_{ij} \omega_{ij} \right) \tag{5.21}$$

$$\mathbf{f}_{ij}^{\text{elastic}} = \mathbf{F}_e \mathbf{M}_i^{-1} \left(\mathbf{r}_{ij} \omega_{ij} \right) \tag{5.22}$$

with $\mathbf{F}_e = -2 V_i \mathbf{J}_i \sigma_i^s$. Note that the matrix product $\mathbf{F}_e \mathbf{M}^{-1}$ is independent of the individual neighbor j and needs to be computed only once for each particle p_i.

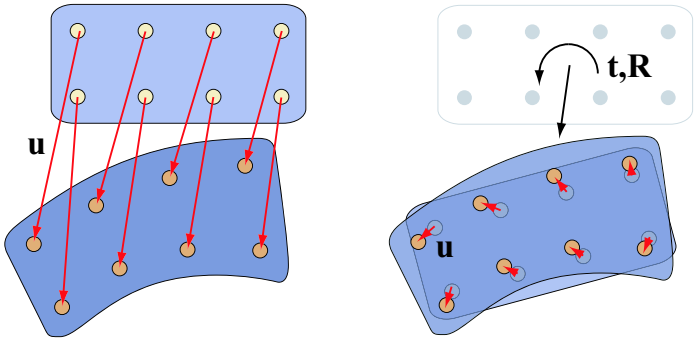


Figure 5.4: Left: a discrete set of displacement vectors $\mathbf{u}_i$ define the deviation of the deformed shape from the rest shape. Right: numerical errors can be reduced by applying an optimal rigid body transformation to the rest shape.

5.4.3 Rigid Transformation of the Rest Shape

If the object departs far from its rest shape, the displacements can become arbitrarily large. This yields two problems. First, computing spatial derivatives of large quantities is numerically unstable. Second, if the moment matrices have bad condition numbers due to volume undersampling, small orientation dependent ghost forces can occur. Both problems are solved by transforming the rest shape after each time step (see Figure 5.4), similar to [TW88]. Unlike their method, we compute the optimal global rotation and translation of the rest shape based on geometric algebra [LFDL98].

Translation. The optimal translation $\mathbf{t} = \overline{\mathbf{x}} - \overline{\mathbf{m}}$ is the difference between the center of mass $\overline{\mathbf{x}}$ in world coordinates and the center of mass $\overline{\mathbf{m}}$ in material coordinates of the particles with

$$\overline{\mathbf{x}} = \sum_j \overline{m}_j \mathbf{p}_j, \tag{5.23}$$

$$\overline{\mathbf{m}} = \sum_j \overline{m}_j \mathbf{m}_j, \tag{5.24}$$

$$\overline{m}_j = m_j / \sum_j m_j, \tag{5.25}$$

where m_j is the mass of a particle p_j and $\mathbf{m}_j$ and $\mathbf{p}_j$ are its position in material and world coordinates, respectively.

Rotation. The optimal rotation $\mathbf{R} = \mathbf{V}\mathbf{U}^T$ is computed in a least-squares sense by computing the singular value decomposition of $\mathbf{S} = \mathbf{U}\mathbf{W}\mathbf{V}^T$, where

$$\mathbf{S} = \sum_j m_j^2 (\mathbf{m}_j - \overline{\mathbf{m}})(\mathbf{p}_j - \overline{\mathbf{x}})^T. \tag{5.26}$$

This linear transformation $(\mathbf{R}, \mathbf{t})$ is applied to the surface representation as well (Section 5.5). Other possibilities on how to transform the rest shape non-rigidly are discussed in Sections 5.10 and 6.4.1.

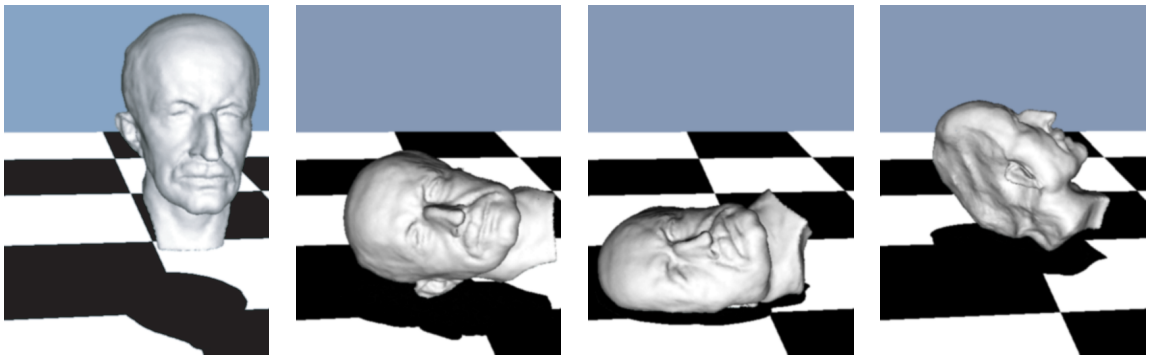


Figure 5.5: Max Planck is elastically and plastically deformed in real-time while switching between material properties on the fly.

5.4.4 Plasticity Model

The standard way in computer graphics for simulating plastic behavior is by using strain state variables [OBH02b]. Every particle p_i stores a plastic strain tensor $\varepsilon_i^{\mathrm{p}}$. The strain considered for elastic forces $\varepsilon_i^{\mathrm{e}} = \varepsilon_i^{\mathrm{s}} - \varepsilon_i^{\mathrm{p}}$ is the difference between measured strain $\varepsilon_i^{\mathrm{s}}$ and the plastic strain. Thus, in case the measured strain is equal to the plastic strain, no forces are generated. Since $\varepsilon_i^{\mathrm{e}}$ is considered constant within one time step, the restoring forces (Equations (5.21) and (5.22)) are computed using $\sigma_i^{\mathrm{e}} = \mathbf{C}\,\varepsilon_i^{\mathrm{e}}$ instead of σ_i^{s}. The plastic strain is updated at every time step Δt according to the following rule

$$\text{if } \|\varepsilon_i^{\mathrm{e}}\| > k^{\mathrm{yield}} \text{ then } \varepsilon_i^{\mathrm{p}} \leftarrow \varepsilon_i^{\mathrm{p}} + \Delta t \cdot k^{\mathrm{creep}} \cdot \varepsilon_i^{\mathrm{e}}, \tag{5.27}$$

where k^{yield} and k^{creep} are material constants. If the 2-norm of the strain exceeds the material yield strength k^{yield}, plasticity starts (Section 2.1.3). In this case, the constant k^{creep} controls how much of the actual deformation is absorbed in the plastic strain state per second.

5.5 Surface Model

For the physical simulation, a coarse sampling of the object with particles is often sufficient to capture the object's elastic behavior. However, the object's surface might be highly detailed with fine geometric features. Therefore, decoupling the surface representation from the physics representation allows efficient simulation of the (coarse) physical model with a highly detailed surface. As a representation we use a point-sampled surface due to its simple structure, which enables efficiently refining the surface for strong deformations (Section 5.5.2). Furthermore, new surface sheets can be created by simply sampling them with surface points, for instance, during fracturing (Section 5.6), for contact handling (Section 5.8.3), and for adapting to topological changes (Section 6.6).

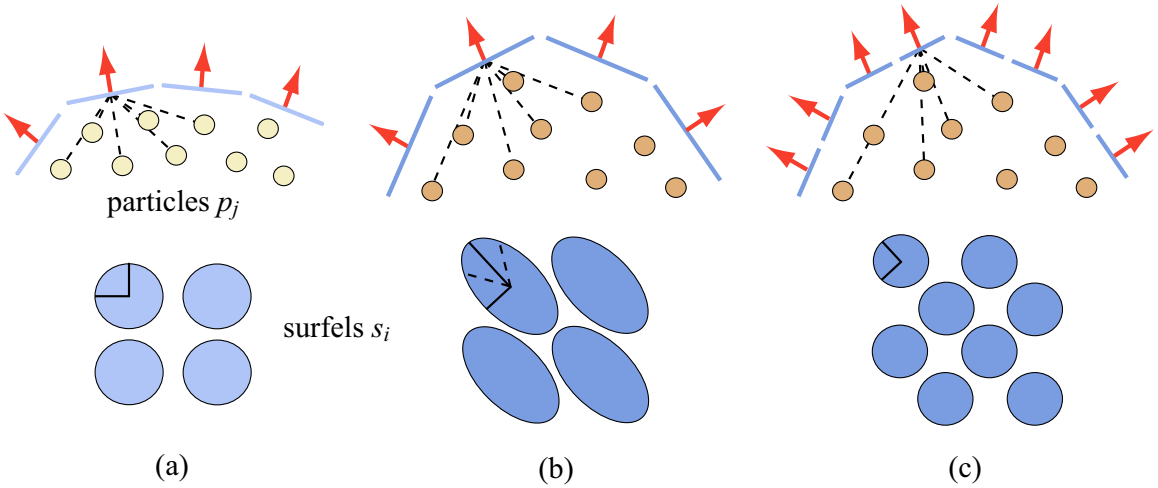


Figure 5.6: Surface animation and refinement. The top row shows the surfels from the side view with the neighboring particles (connect by dashed black lines), the bottom row shows the surfels from the front view. (a) surfels/particles in material coordinates. (b) surfels/particles in world coordinates after a deformation. (c) surfels/particles in world coordinates after splitting.

5.5.1 Surface Animation

We start with an object whose surface $\mathcal{S}$ is sampled with oriented surface elements (*surfels*) $s \in \mathcal{S}$. A surfel s has a position $\mathbf{m}_s$ and two orthogonal tangent vectors $\mathbf{t}_m^1$ and $\mathbf{t}_m^2$, which define the surfel center and the (elliptic) area covered by s in *material coordinates*, respectively. The surfel normal is computed as $\mathbf{t}_m^1 \times \mathbf{t}_m^2$. The surfel position in *world coordinates* is denoted as $\mathbf{s}$, and the tangent axes as $\mathbf{t}_s^1$ and $\mathbf{t}_s^2$.

For animating $\mathcal{S}$, i.e., computing the surfel positions in world coordinates, we make use of the continuous displacement vector field $\mathbf{u}(\mathbf{m})$ defined by the particles. This means that the surfels are advected along with the particles (see Figure 5.6). The displacement vector $\mathbf{u}(\mathbf{m})$ at a known surfel position $\mathbf{m}_s$ is computed from the displacements $\mathbf{u}_j$ of the neighboring particles p_j. For this we need to define a smooth displacement vector field in $\mathbb{R}^3$ that is invariant under rigid transformations. This is achieved by using a first order moving least-squares approximation (Section 2.3.4). However, we have already obtained such an approximation of $\nabla \mathbf{u}_j$ for the particles (see Section 5.4.1), which is reused here. Thus, the displacement vector $\mathbf{u}(\mathbf{m})$ is approximated as

$$\langle \mathbf{u}(\mathbf{m}) \rangle = \frac{\sum_j \omega^{\text{poly}}(\mathbf{m} - \mathbf{m}_j, h_{\mathcal{P}}) \left(\mathbf{u}_j + \nabla \mathbf{u}_j^T (\mathbf{m} - \mathbf{m}_j) \right)}{\sum_j \omega(\mathbf{m} - \mathbf{m}_j, h_{\mathcal{P}})}, \tag{5.28}$$

where we use the smoothing kernel ω^{poly} as a weighting function (see Equation (4.9)). The $\mathbf{u}_j$ are the displacement vectors of particles at $\mathbf{m}_j$ that are within

a distance $h_{\mathcal{P}}$ to $\mathbf{m}_s$.

We apply Equation (5.28) to both the surfel center and its tangent axis. This gives the surfel position $\mathbf{s}$ in world coordinates as $\mathbf{s} = \mathbf{m}_s + \mathbf{u}(\mathbf{m}_s)$, and the tangent vector in world coordinates as $\mathbf{t}_s^1 = (\mathbf{u}(\mathbf{m}_s + \mathbf{t}_m^1) - \mathbf{u}(\mathbf{m}_s))$ and analogously for $\mathbf{t}_s^2$.

5.5.2 Surface Refinement

When the surface deforms surfels are stretched or compressed. For large deformations this can yield distortions due to the (local) undersampling of the surface [Pau03]. Thus, the sampling of the surface needs to be refined to maintain a high quality surface. Initially, the tangent axes in world coordinates $\mathbf{t}_s^1$ and $\mathbf{t}_s^2$ of a surfel s are equal to the orthogonal tangent axes in material coordinates. During a deformation, the world coordinates are shifted. Thus, if a surface is stretched or compressed, the tangent axes in world coordinates are no longer orthogonal to each other and their length changes. A measurement of this local distortion can be found by looking at the first fundamental form at $\mathbf{s}$, which is defined as

$$\begin{bmatrix} \mathbf{t}_s^1 \cdot \mathbf{t}_s^1 & \mathbf{t}_s^1 \cdot \mathbf{t}_s^2 \\ \mathbf{t}_s^1 \cdot \mathbf{t}_s^2 & \mathbf{t}_s^2 \cdot \mathbf{t}_s^2 \end{bmatrix}. \tag{5.29}$$

The eigenvalues of the matrix in Equation (5.29) give the minimum and maximum stretch factors and the corresponding eigenvectors define the principal directions of the stretching. Thus, the ratio of the two eigenvalues is a measurement of the local anisotropy. When this ratio is larger than a threshold, the surfel is split into two new surfels which are positioned on the main axis of the ellipse as shown in Figure 5.6.

Ideally, if an object is first stretched and then compressed to its original shape, the surface should not change. We use a simple approach to achieve this. Instead of deleting a split surfel s_i, it is kept in material coordinates. If both children of this surfel are compressed (the ratio of the eigenvalues is smaller than one), also the world coordinates of s_i are computed. If s_i does not fulfill the splitting criterion, its children are simply deleted and the parent surfel is used again instead. Note that the overhead is small because for inactive surfels only the rigid transformation of the rest shape needs to be applied to them (see Section 5.4.3).

5.6 Fracture Model

The previous section described a framework for the animation of elasto-plastic materials. In the following, we will discuss how this framework can be extended for simulating fracturing solids. Central to the method is a highly dynamic surface and volume sampling method that supports arbitrary crack initiation, propagation, and termination, while avoiding many of the stability problems of traditional

Figure 5.7: Brittle fracture of a hollow stone sculpture. Forces acting on the interior create stresses which cause the model to fracture and explode. Initial/final sampling: 4.3k/6.5k particles, 249k/310k surfels, 22s/frame.

mesh-based techniques. Advancing crack fronts are modeled explicitly and associated fracture surfaces are embedded in the simulation volume. When cutting through the material, crack fronts directly affect the coupling between particles, requiring a dynamic adaptation of the particle shape functions. Complex fracture patterns of interacting and branching cracks are handled using a small set of topological operations for splitting, merging, and terminating crack fronts. This allows continuous propagation of cracks with highly detailed fracture surfaces, independent of the spatial resolution of the particles, and provides effective mechanisms for controlling fracture paths. The method is applicable for a wide range of materials, from stiff elastic to highly plastic objects that exhibit brittle and/or ductile fracture.

5.6.1 Introduction

Physically, fracturing occurs when the internal stresses and the resulting forces are so large that the interatomic bounds cannot hold the material together anymore. Fracturing has been studied extensively in the physics and mechanics literature. However, due to the complexity of the problem, the studies and simulations usually deal only with "simple" fractures such as the creation or propagation of a single crack. In computer graphics, we often trade physical accuracy for visual realism. By simplifying the physical model, realistic animations of very complex fractures such as the shattering of glass into hundreds of pieces can be achieved. However, changing the topology of a simulated object is challenging for both the animation of the volume and the surface. When a solid fractures, the surface needs to adapt to the cracks that propagate through the volume of the solid. To achieve a high degree of visual realism, cracks should be allowed to start anywhere on the surface and move in any direction through the volume. Furthermore, cracks might branch into several cracks, or different cracks can merge to a single crack within the solid. While fracturing, not only the topology of the surface changes, but also the discontinuities introduced by the cracks in the volume have to be modeled accordingly to achieve physically plausible fracture behavior. The fracturing characteristics depend on the material. We differentiate between ductile and brittle fracture. While brittle material splits without experiencing significant irre-

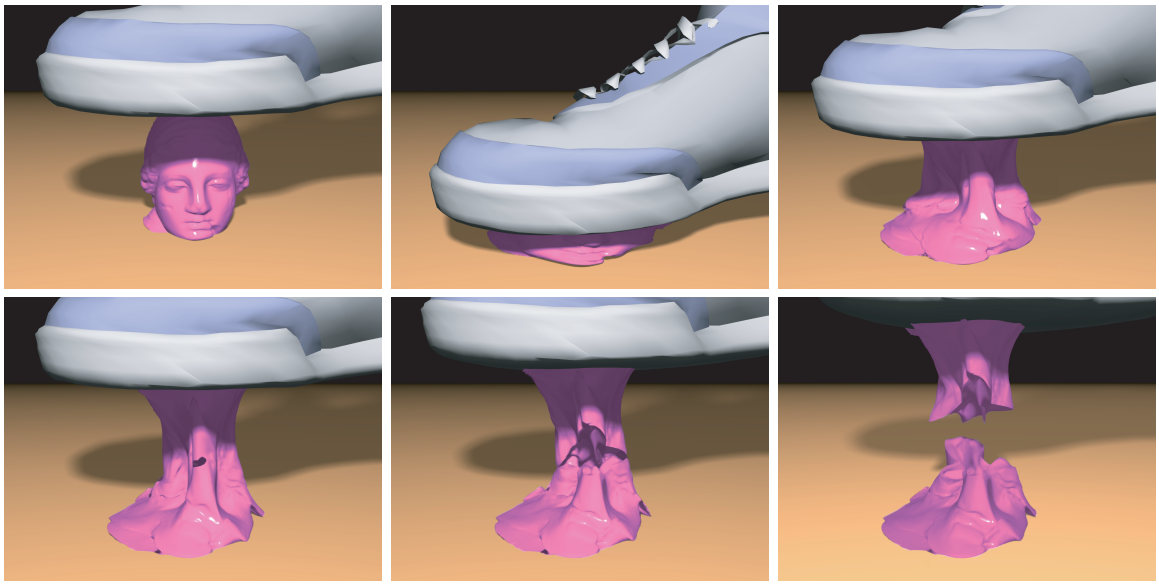


Figure 5.8: Highly plastic deformations and ductile fracture. The bubble gum like material is first deformed beyond recognition. It is then stretched until the stress in the material is too high and it fractures along a complex fracture surface. Initial/final sampling: 2.2k/3.3k particles, 134k/144k surfels, 2.4s/frame.

versible deformation (i.e., only elastic deformation), ductile material experience some amount of plastic deformation before fracture [OBH02a]. Two examples for brittle and ductile materials are shown in Figure 5.7 and Figure 5.8. A force acting on the hollow stone sculpture in Figure 5.7 causes the model to explode. Due to the simulated brittle material this results in a shattering of the object into pieces. Figure 5.8 shows ductile fracture of a highly plastic bubble gum like material which is deformed beyond recognition before splitting along a single complex fracture surface.

5.6.2 Modeling Discontinuities

We will first discuss how the discontinuity can be modeled that is introduced by a propagating crack into the domain of a simulated solid. For that, the so-called visibility criterion [BLG94] can be used where particles are allowed to interact with each other only if they are not separated by a surface. This is done by testing if a ray connecting two particles intersects the boundary surface.

To see what happens when we use the visibility criterion we look at the discretization $\langle \mathbf{u} \rangle$ of the continuous displacement field $\mathbf{u}$. As shown in Section 2.3.4, we can approximate $\mathbf{u}$ in the form

$$\langle \mathbf{u}(\mathbf{m}) \rangle \approx \sum_j \Phi_j(\mathbf{m}_j)\mathbf{u}_j, \tag{5.30}$$

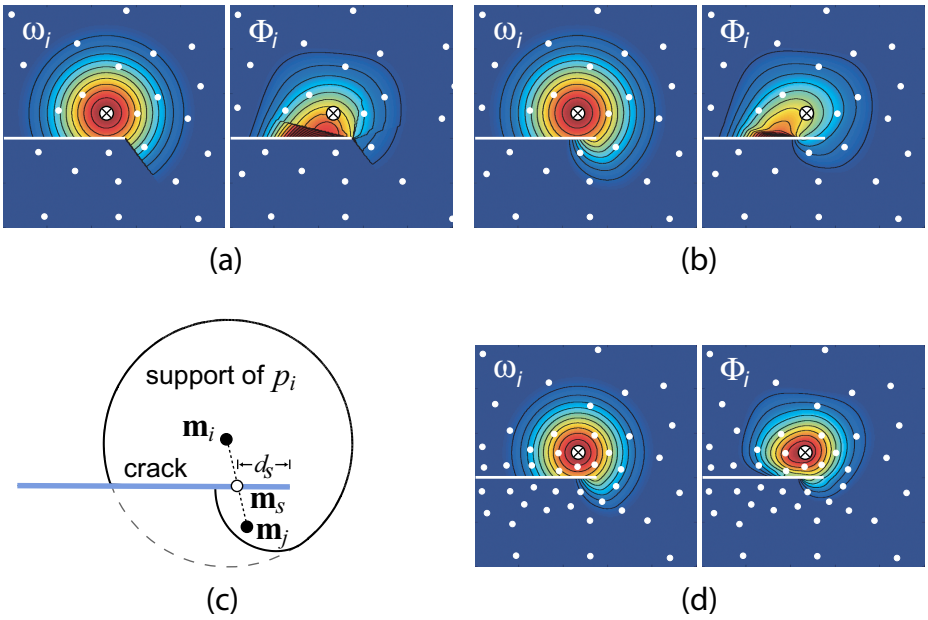


Figure 5.9: Comparison of visibility criterion (a) and transparency method (b) for an irregularly sampled 2D domain. The effect of a crack, indicated by the horizontal white line, on weight function ω_i and shape function Φ_i is depicted for particle p_i marked by the cross. A schematic view of the transparency method is shown in (c) and the effect of dynamic upsampling is illustrated in (d).

where $\Phi_j(\mathbf{m}_j)$ is the shape function of a particle p_j with position $\mathbf{m}_j$ in material coordinates and $\mathbf{u}_j$ is the field value at $\mathbf{m}_j$, i.e., in our case the displacement of p_j into world coordinates $\mathbf{p}_j$. As described in Section 2.3.4, an approximation with consistency order n can be achieved by using *moving least-squares (MLS)* interpolants. Given a complete polynomial basis $\mathbf{b}(\mathbf{m}) = [1\ \mathbf{m}\ \ldots\ \mathbf{m}^n]^T$ of order n and a weight function ω, the meshless shape functions are derived as

$$\Phi_i(\mathbf{m}) = \mathbf{b}^T(\mathbf{m})\mathbf{M}^{-1}(\mathbf{m})\mathbf{b}(\mathbf{m})\omega_i(\mathbf{m}-\mathbf{m}_i,h_i), \tag{5.31}$$

where $\mathbf{M}$ is the moment matrix defined as

$$\mathbf{M}(\mathbf{m}) = \sum_j \mathbf{b}(\mathbf{m}_j)\mathbf{b}^T(\mathbf{m}_j)\omega_j(\mathbf{m}-\mathbf{m}_j,h_j), \tag{5.32}$$

see Section 2.3.4 for details.

Figure 5.9 (a) shows the weight and shape functions when using the visibility criterion. The crack not only introduces a discontinuity along the crack surface, but also undesirable discontinuities of the shape functions within the domain. The transparency method proposed by Organ et al. [OFTB96] alleviates potential stability problems due to these discontinuities. The idea is to make the crack more transparent closer to the crack front. This allows partial interaction of particles in the vicinity of the crack front. Suppose the ray between two particles p_i and p_j

intersects a crack surface at a point $\mathbf{m}_s$ (Figure 5.9 (c)). Then the weight function ω_i (and similarly for ω_j) is adapted to

$$\tilde{\omega}_i(\mathbf{r}_{ij}, h_i) = \omega_i\left(\mathbf{r}_{ij}\left(1 + h_i\left(\frac{2d_s}{k^{\text{opacity}} h_i}\right)^2\right), h_i\right) \tag{5.33}$$

where d_s is the distance between $\mathbf{m}_s$ and the closest point on the crack front, and k^{opacity} controls the opacity of the crack surfaces. Effectively, a crack passing between two particles lengthens the interaction distance of the particles until eventually, in this adapted distance metric, the particles will be too far apart to interact. As shown in Figure 5.9 (b) this method avoids the discontinuities of the shape functions within the domain and thus leads to increased stability.

To compute the ray intersection with the crack surface sampled with surfels, the implicit surface defined by the surfels is used (see Section 2.5.1). The intersection point $\mathbf{m}_s$ on this surface is then found using Brent's method as described in Section 2.4.2.

5.6.3 Fracture Surface Model

Introducing cuts into the model exposes interior parts of the solid which need to be bounded by new surface sheets. Previous approaches based on FEM define fracture surfaces using faces of the tetrahedral elements, which requires complex dynamic remeshing to avoid unnaturally coarse crack surfaces [OH99]. To simplify the topological complexity and avoid stability problems during the simulation, mesh-based approaches impose restrictions on where and how the material can fracture. These restrictions can be lifted by embedding a surface and explicitly creating new fracture surface sheets whenever the material is cut. In this section we will describe how we extend our point-based surface model described in Section 5.5 for creating these surface sheets. We will show that this extension is simple and efficient, since no explicit connectivity information needs to be maintained between surfels. Sharp creases and corners are represented implicitly as the intersection of adjacent surface sheets using a CSG method. The precise location of crease lines is evaluated at render time (cf. Figure 5.10), avoiding costly surface-surface intersection calculations during simulation.

A crack consists of a crack front and two separate surface sheets which are connected at the front to form a sharp crease. The crack front itself is defined by a linear sequence of crack nodes $\mathbf{c}_1, \ldots, \mathbf{c}_n$. Surfels are added continuously to the fracture surfaces while propagating through the material. For surface cracks the end nodes of the front lie on a boundary surface or a fracture surface of a different crack. Interior cracks have circularly connected crack fronts, i.e., the two end nodes $\mathbf{c}_1$ and $\mathbf{c}_n$ coincide (cf. Figures 5.11 and 5.13).

To animate the boundary surface of the solid, the free-form deformation approach described in Section 5.5.1 is used. To ensure that the displacement field

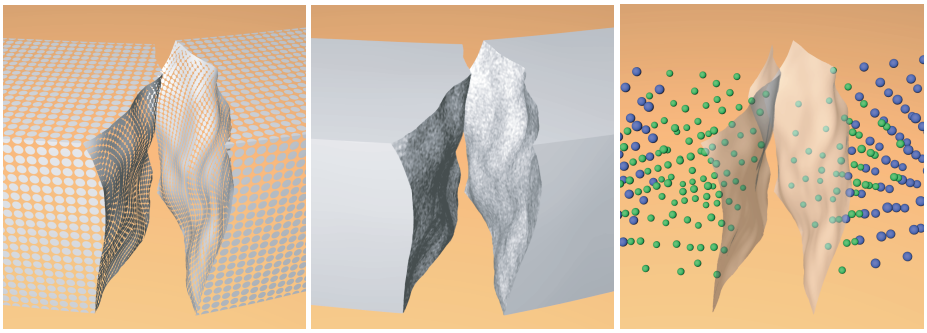


Figure 5.10: Surfels are clipped to create sharp creases with dynamically created fracture surfaces, whose visual roughness is controlled using 3D noise functions for bump mapping. The sampling of the simulation domain is shown on the right, where green spheres denote resampled particles.

is smooth at the crack front, the transparency weights $\tilde{\omega}$ described above are also used in Equation (5.28) for the displacement computation of the surfels (cf. Figure 5.12). Because the changes of the transparency weights are localized to a small region around the crack front, only a small fraction of the weights need to be updated in every time step, leading to an efficient implementation.

5.6.4 Crack Initiation and Propagation

Crack initiation is based on the stress tensor σ^e. A new crack is created where the maximal eigenvalue of σ^e exceeds the threshold for tensile fracture (opening mode fracture [And95]). This condition is evaluated for all particles. To allow crack initiation anywhere on the surface or in the interior of the model, a stochastic scheme can be applied to initiate crack fronts. A random set of surface and interior sample points are created and the stress tensor at these points is evaluated using weighted averaging from adjacent particles. The inherent smoothing is usually desired to improve the stability of the crack propagation. If a crack front is initiated at one of these spatial locations, the fracture thresholds of all neighboring samples are increased to avoid spurious branching.

A new crack is initialized with three crack nodes, each of which carries two surfels with identical position and tangent axes, but opposing normals. These surfels form the initial crack surfaces which will grow dynamically as the crack propagates through the solid (Figure 5.11). Crack propagation is determined by the propagation vectors $\mathbf{d}_i = k^{\text{prop}} \lambda_i (\mathbf{e}_i \times \mathbf{t}_i)$, where λ_i is the maximal eigenvalue of the stress tensor at $\mathbf{c}_i$, and $\mathbf{e}_i$ is the corresponding eigenvector. The vector $\mathbf{t}_i$ approximates the tangent of the crack front as $\mathbf{t}_i = (\mathbf{c}_{i+1} - \mathbf{c}_{i-1}) / \|\mathbf{c}_{i+1} - \mathbf{c}_{i-1}\|$, where $\mathbf{c}_0 = \mathbf{c}_1$ and $\mathbf{c}_{n+1} = \mathbf{c}_n$ for surface cracks. The parameter k^{prop} depends on the material and can be used to control the speed of propagation. The new position of a crack node $\mathbf{c}_i$ at time $t + \Delta t$ is then computed as $\mathbf{c}_i + \Delta t \mathbf{d}_i$, where Δt is the sim-

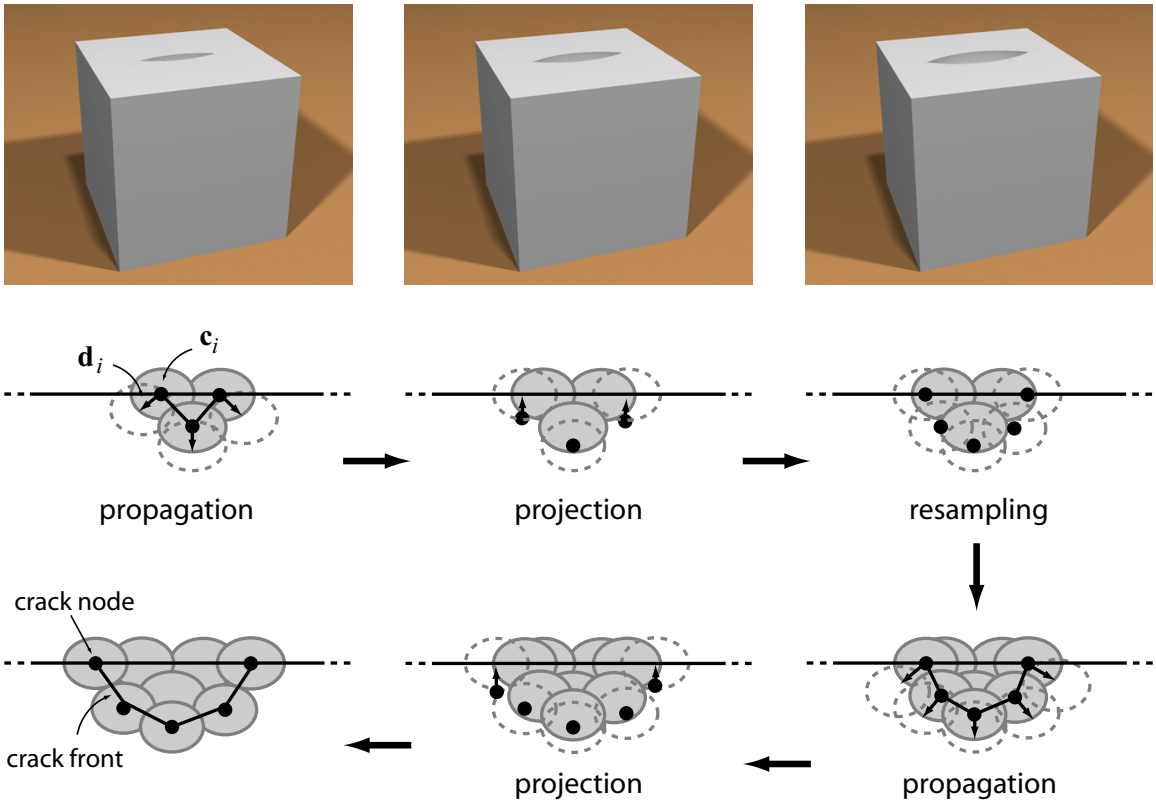


Figure 5.11: Front propagation and fracture surface sampling. The upper row shows a top view of an opening crack, the lower part shows a side view of a single fracture surface. After propagating the crack nodes $\mathbf{c}_i$ according to $\mathbf{d}_i$, end nodes are projected onto the surface. If necessary, the front is resampled and new surfels are added to the fracture surface sheets.

ulation time step. Additionally, the end nodes of surface cracks are projected back onto the surface that they originated from using the projection method described in Section 2.5.1. Since propagation alters the spacing of crack nodes along the front, the sampling resolution of the crack nodes are adjusted dynamically after each propagation step. If two adjacent crack nodes are further apart than the radius of their associated surfels, a new node is inserted using cubic spline interpolation to determine the new node's position. Redundant crack nodes are removed when the distance to the immediate neighbors becomes too small. Fracture surface sheets are sampled by inserting new surfels if the propagation distance exceeds the surfel radius, indicating that a hole would appear in the surface. This spatially (along the crack front) and temporally (along the propagation vectors) adaptive sampling scheme ensures uniformly sampled and hole-free crack surfaces (cf. Figure 5.11).

During crack propagation, the simulation is adjusted automatically to the newly created fracture surfaces by adapting the shape functions using the transparency method described above. The transparency weight $\tilde{\omega}_i(\mathbf{r}_{ij}, h_i)$ for a pair of particles

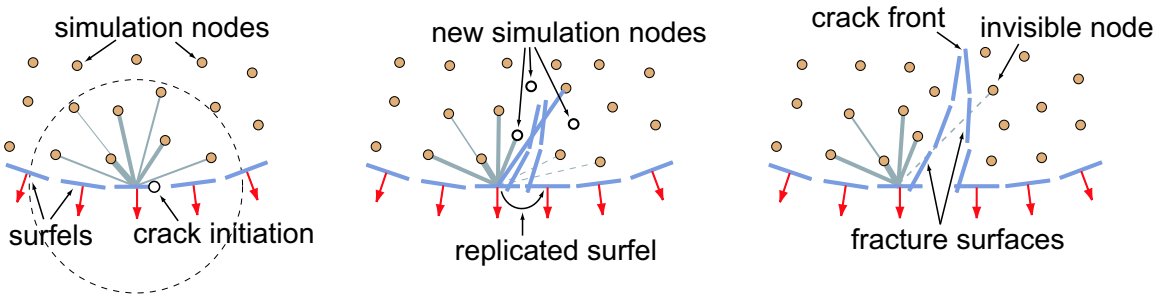


Figure 5.12: Transparency weights for embedding surfels in the simulation domain. The thickness of the lines indicates the influence of a particle on the displacement of a surfel. During crack propagation, new surfels and particles are created using dynamic resampling as described below.

is adapted by computing the intersection point on the fracture surface of the ray connecting the two particles (see Section 5.6.2). The distance d_s to the crack front is approximated as the shortest Euclidean distance to the line segments defined by adjacent crack nodes. To avoid stability problems with curved fracture surfaces, weights are allowed to only decrease from one time step to the next.

5.6.5 Topology Control

The major challenge when explicitly modeling fracture surfaces is the efficient handling of all events that affect the topology of the boundary surface and the simulation domain. Apart from crack initiation, three fundamental events are sufficient to describe the often intricate constellations that occur during fracturing: *Termination*, *splitting*, and *merging* of crack fronts:

- A crack is terminated if the crack front has contracted to a single point.
- Splitting occurs when a crack front penetrates through a surface as shown in Figure 5.13 (a). The signed distance of a crack node to a surface sheet can be estimated using the projection operator described in Section 2.5.1. A splitting event is initiated when a sign change occurs from one time step to the next. The front is split at the edges that intersect the surface, discarding all nodes that are outside the solid, except the ones that are connected to an interior node. These nodes become new end nodes by moving them to the intersection point with the surface. As shown on the left of Figure 5.13 (a), a surface crack is split into two new crack fronts that share the same crack surfaces, i.e., independently add surfels to the same fracture surface sheets during propagation. An interior crack becomes a surface crack after splitting, as illustrated on the right.
- A merging event is triggered when two surface end nodes of two crack fronts meet by creating the appropriate edge connections (Figure 5.13 (b)). Two

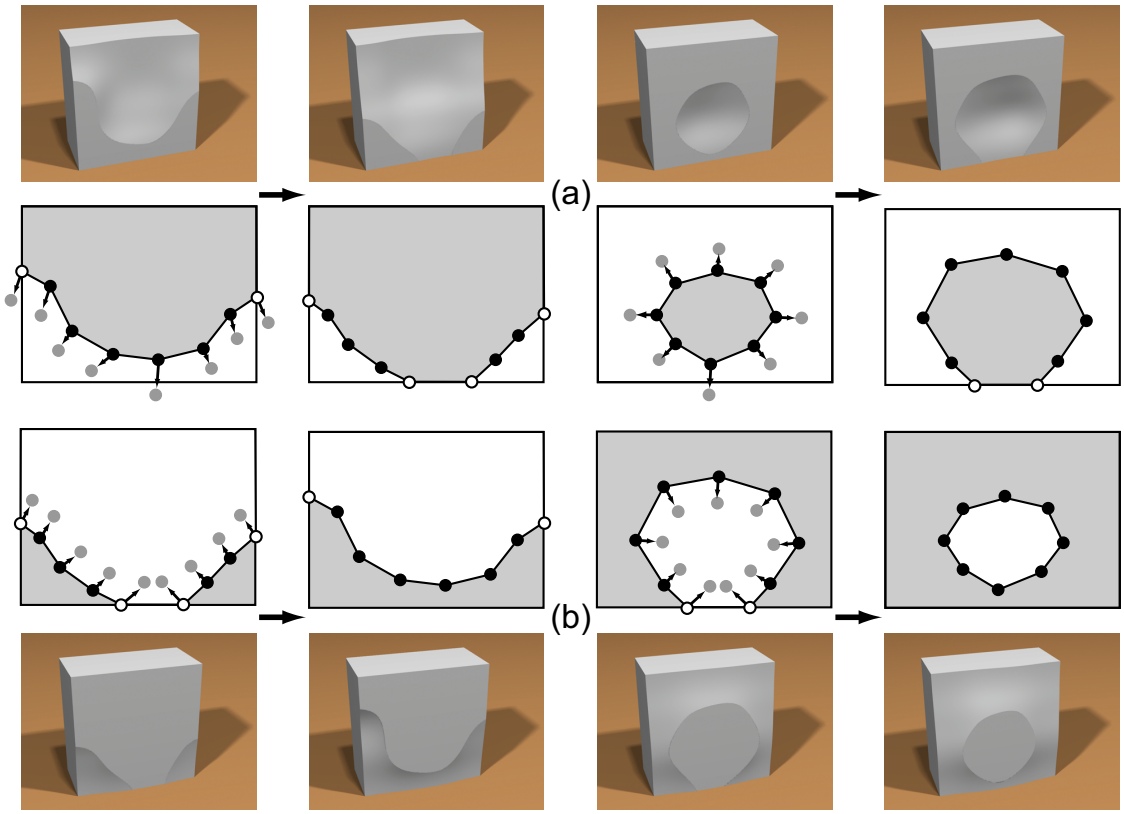


Figure 5.13: Topological events during crack propagation. (a) Splitting, (b) merging. The top and bottom rows show a cutaway view with one crack surface exposed. The sketches in the center rows show this fracture surface in gray, end nodes of crack fronts are indicated by white dots.

surface cracks are merged into a single surface crack (left), while a circular front is created if the two end nodes are from the same crack front (right). Typically, when cracks merge, their fracture surfaces create a sharp corner, so we maintain separate fracture surface sheets that intersect to create a crease.

As can be seen in Figure 5.13, splitting and merging are dual to each other. The former introduces two new end nodes, whereas the latter decreases the number of end nodes by two. Similarly, crack initiation and termination are dual topological operations. Note that the intersection of two crack fronts at interior nodes is handled automatically by first splitting both fronts and then merging the newly created end nodes.

One useful technique to improve the stability of the simulation is *snapping*. Snapping guarantees that problematic small features, such as tiny fragments or thin slivers, do not arise. It works by forcing nodes very near other nodes to become coincident and projects nodes onto the surface if they are very close to it

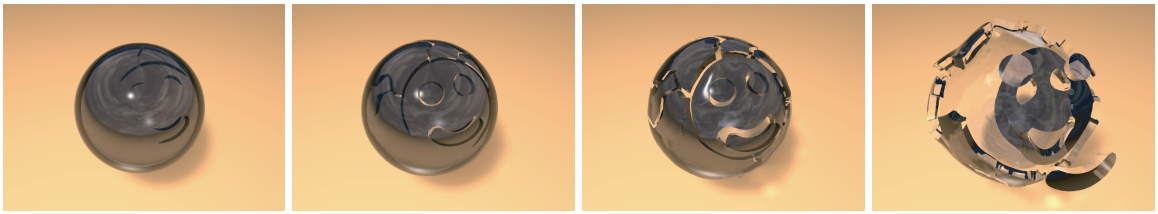


Figure 5.14: Controlled fracture. While the sphere blows up it fractures along the prescribed smiley face. Initial/final sampling: 4.6k/5.8k particles, 49k/72k surfels, 6s/frame.

to ensure that any features present are of size comparable to the local node spacing. Similar methods have been proven to guarantee topological consistency with the ideal geometry in other settings [GM95]. Specifically, when a front intersects a surface, all crack nodes are projected that are within snapping distance d to the surface onto the surface. This avoids fragmenting the front into small pieces that would be terminated anyway within a few time steps. Furthermore, fronts are merged when the end nodes are within distance d by moving both end nodes to their average position. This avoids small slivers of material to be created, which would require adding a significant number of new particles to the model (see Section 5.7). Similarly, the intersection of two crack fronts can lead to multiple splitting and merging events, which are combined into a single event to avoid the overhead of creating and subsequently deleting many small crack fronts. Snapping can also be applied to front termination, where a crack front is deleted when all its nodes are within distance d from each other.

5.6.6 Fracture Control

The course of the simulation can be influenced by specifying material properties. However, often direct control over the fracture behavior is crucial, especially in production environments and interactive applications where the visual effect is usually more important than physical accuracy. By exploiting the explicit point-based representation of the fracture surfaces, the fracture framework can be extended to support precise control of where and how a model fractures. One possibility is to use a painting interface (see e.g. [ZPKG02, AWD$^+$04]) that allows fast prototyping of fracture simulations by prescribing fracture patterns directly on the object boundary. The user can paint arbitrary networks of cracks on the surface and explicitly specify stress thresholds for these cracks. Additionally, a propagation history can be used to control the propagation of cracks through the material. The adjusted propagation vector at time t is computed as the weighted average $\overline{\mathbf{d}}_i^t = k^{\text{hist}}\mathbf{d}_i^{t-\Delta t} + (1-k^{\text{hist}})\mathbf{d}_i^t$, where $k^{\text{hist}} \in [0,1]$ is the history factor. A purely stress-based propagation is achieved for $k^{\text{hist}} = 0$, while $k^{\text{hist}} = 1$ yields purely geometric cracks and fracture surfaces. Other possibilities include volumetric tex-

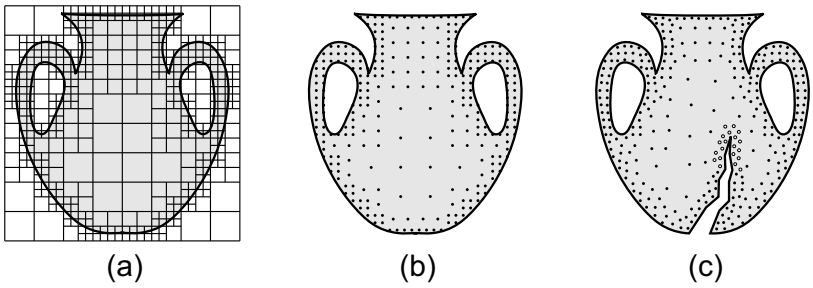


Figure 5.15: Volumetric sampling: (a) octree decomposition, (b) initial adaptive octree sampling, (c) dynamic resampling during fracturing.

tures for adjusting the fracture thresholds within the material, and pre-scoring techniques, where the stress tensor is modified according to an embedded level set function [MBF04]. Figure 5.14 shows an example of an explicitly controlled fracture, using a combination of crack painting, propagation history, and adaptive fracture thresholds.

5.7 Volumetric Sampling

One of the main advantages of meshless methods lies in the fact that they support simple and efficient sampling schemes. Initially, the volume Ω bounded by a surface $\mathcal{S}$ of an object Γ is discretized by sampling Ω with particles. Similar to adaptive finite element meshing, we want a higher particle density close to the boundary surface and fewer particles towards the interior of the solid. An appropriate sampling of the particles can be computed, for instance, using a balanced octree hierarchy as shown in Figure 5.15. Starting from the bounding box of $\mathcal{S}$, a cell of the octree is recursively refined, if it contains parts of $\mathcal{S}$. The final number of particles is controlled by prescribing a maximum octree level at which the recursive refinement is stopped. Given this adaptive decomposition, a particle is created at each octree cell center that lies within Ω.

During simulation, the discretization of the simulation domain needs to be adjusted dynamically. Without dynamic resampling, frequent fracturing would quickly degrade the numerical stability of the simulation even for an initially adequately sampled model. New particles need to be inserted in the vicinity of the crack surfaces and in particular around the crack front. At the same time, strong deformations of the model can lead to a poor spatial discretization of the simulation volume, which also requires a dynamic adaptation of the sampling resolution. This is particularly important for highly plastic materials, where the deformed shape can deviate significantly from its original configuration.

A simple local criterion can be used to determine under-sampling at a particle

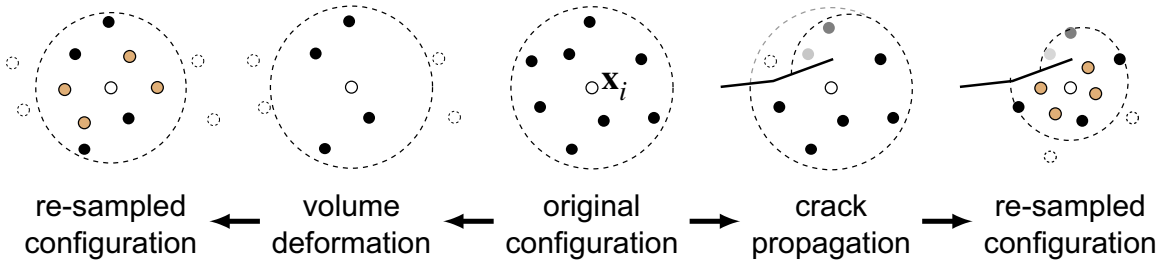


Figure 5.16: Dynamic resampling at the particle p_i due to strong deformation (left) and fracturing (right).

p_i. Let

$$w_i = \sum_j \tilde{\omega}_i(\mathbf{r}_{ij}, h_i)/\omega_i(\mathbf{r}_{ij}, h_i) \tag{5.34}$$

be the normalized sum of transparency weights (see Section 5.6.2). Without visibility constraints, w_i is simply the number of particles in the support of p_i. During simulation w_i decreases, if fewer neighboring particles are found due to strong deformations, or if the transparency weights become smaller due to a crack front passing through the solid. If w_i drops below a threshold $w_{\min}$, new particles are inserted within the support radius of p_i (see Figure 5.16). For the dynamic up- and downsampling and robust interaction between particles with different support radii the multiresolution approach described in Section 4.5 can be exploited, where $w_i \leq w_{\min}$ is used as a splitting criterion.

To prevent excessive resampling for particles very close to a fracture boundary, particle splitting is restricted by prescribing a minimal particle support radius h. Note that resampling due to fracturing is triggered by the crack nodes passing through the solid, similar to adapting the visibility weights (see Section 5.6.3). Performing these checks comes essentially for free, since all the required spatial queries are already carried out during visibility computation. Figure 5.15 (d) and Figure 5.10 illustrate the dynamic adaptation of the sampling rates when fracturing. The effect on the shape functions is shown in Figure 5.9 (d).

5.8 Contact Model

Soft objects deform due to external forces exerted during collisions with the environment or with other objects such as a user guided tool. Therefore, both the collision detection algorithm as well as the collision response model play a central role in the simulation of deformable objects.

In this section the framework for elasto-plastic and fracturing objects Γ_i, where the object volume Ω_i is sampled with particles p and the surface $\mathcal{S}_i$ is represented by surfels s (see Section 5.4) is extended for collision detection and response. Collisions are stably detected using the implicit surface representation described

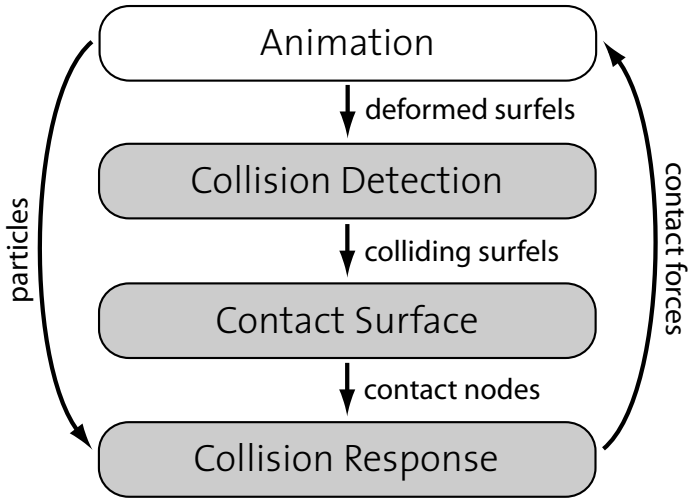


Figure 5.17: Overview of the contact handling pipeline (gray shaded). After the physics and surface animation, the colliding surfels are detected and the contact surface is computed. The collision response forces are computed for the contact nodes on the contact surface and distributed to the particles.

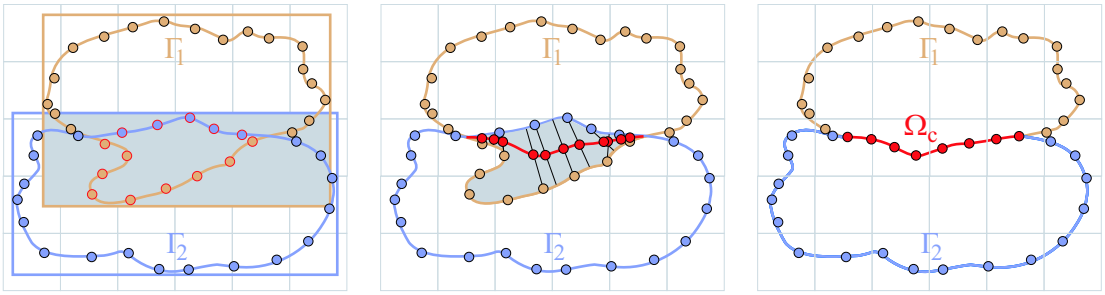
in Section 2.5.1. In case of a collision, response forces are computed for penetrating surfels and distributed to particles. The decoupling of collision handling and deformation allows for a very stable collision response while maintaining interactive update rates of the dynamic simulation for environments with moderate complexity.

5.8.1 Overview

Figure 5.17 gives an overview of our collision detection and response algorithm which is built on top of the deformation and fracturing framework described above. After each animation step, the collision detection algorithm gets the point-sampled surfaces as input and computes the colliding surfels (Section 5.8.2). From the colliding surfels, a contact surface is computed which resolves the intersecting surfaces in a plausible way (Section 5.8.3). For the surfels on the contact surface (called *contact nodes*), penalty forces are computed and distributed to the neighboring particles such that momentum of the system is preserved (Section 5.8.4).

5.8.2 Collision Detection

Collision detection for point-based objects amounts to finding surfels that are inside other surfel-bound objects. Bounding volume hierarchies are a means to reduce the number of primitives (surfels in our case) to be tested for intersection. However, because we deal with highly deformable models, the update of such a



(a) (b) (c)

Figure 5.18: (a) Detection of colliding surfels by first intersecting the bounding boxes. The points in the shadowed region are collision candidates. Colliding points are outlined in red. (b) In the case where both objects have the same stiffness, the contact surface is the middle of the intersecting volume, shown as red line. It is initialized by moving the colliding points onto it. (c) The final surfaces after resampling the contact surface.

hierarchy is often very costly. Therefore, we approximate the objects by only one bounding volume to efficiently discard collisions if the bounding volumes do not overlap.

We choose axis aligned bounding boxes (AABBs) [van97] because they can be computed very efficiently. Furthermore, the intersection volume of two overlapping AABBs is again an AABB, here denoted with $\mathcal{B}$. Only the surfels s_i that are inside $\mathcal{B}$ are candidates for a collision. These surfels are efficiently found by first inserting all surfels into a uniform spatial grid and then intersecting $\mathcal{B}$ with the cells of this grid (see Figure 5.18 (a)). The grid is used afterwards as a search data structure for range and neighbor queries (see Section 7.1.1). A similar approach is suggested by Teschner et al. for collision detection of tetrahedral meshes [THM+03].

In a next step, it is determined whether the collision candidates in $\mathcal{B}$ are actually penetrating other objects. For that, the implicit surface representation Ψ defined by the surfels as described in Section 2.5.1 is used. With such an implicit representation it can be efficiently computed if the center $\mathbf{s}_i$ of a surfel s is inside an object Γ as follows:

$$\mathbf{s}_i \cap \Gamma \Longleftrightarrow \left(\mathbf{s}_i - \psi^{\text{orth}}(\mathbf{s}_i)\right) \cdot \mathbf{n}_{\psi^{\text{orth}}(\mathbf{s}_i)} < 0, \tag{5.35}$$

where $\psi^{\text{orth}}(\mathbf{s}_i)$ is the orthogonal projection operator described in Section 2.5.1, which gives the projected position of $\mathbf{s}_i$ onto Ψ, and $\mathbf{n}_{\psi^{\text{orth}}(\mathbf{s}_i)}$ is the normal vector of Ψ at $\psi^{\text{orth}}(\mathbf{s}_i)$ (Section 2.5.1). We define that a surfel is penetrating an object if its center is inside the object.

For simplicity, we assume in the following sections that we have only two possibly colliding objects Γ_1 and Γ_2. In Section 5.8.5 the contact handling for an arbitrary number of objects is discussed.

5.8.3 Contact Surface

At the time of collision two colliding surfaces intersect. We get a visually consistent contact surface by assuming that the surface is elastic. Thus, we *temporarily* recompute the intersecting surfaces such that they touch. The displacement of the surface to the contact surface is used for computing the collision response, as described in the next section, and for rendering. Thus, in the next time step the contact surface is discarded, and the surface before contact handling is used for the further animation of the surface.

Assume that we have two intersecting objects $\Gamma_1 = \{\mathcal{S}_1, \Omega_1\}$ and $\Gamma_2 = \{\mathcal{S}_2, \Omega_2\}$, where $\mathcal{S}_i$ is the surface and Ω_i the volume of Γ_i. We aim to efficiently compute a reasonable contact surface $\mathcal{S}_c$ that lies in the intersection volume Ω_c of Ω_1 and Ω_2, i.e., $\Omega_c = \Omega_1 \cap \Omega_2$.

Using the orthogonal projection operator $\psi^{\text{orth}}(\mathbf{x})$ described in Section 2.5.1 we can efficiently compute the closest distance of a point $\mathbf{x}$ to the implicit surface representation Ψ_i of Γ_i as $\left\|\psi_i^{\text{orth}}(\mathbf{x}) - \mathbf{x}\right\|$. We define a function

$$F(\mathbf{x}) = \frac{1}{k_1^c + k_2^c}\left(k_1^c \left\|\psi_1^{\text{orth}}(\mathbf{x}) - \mathbf{x}\right\| - k_2^c \left\|\psi_2^{\text{orth}}(\mathbf{x}) - \mathbf{x}\right\|\right). \tag{5.36}$$

We use this function to define the contact surface $\mathcal{S}_c$ implicitly as

$$\mathcal{S}_c = \{\mathbf{x} \in \Omega_c \mid F(\mathbf{x}) = 0\}, \tag{5.37}$$

where k_1^c and k_2^c are constants depending on the surface material. If k_1^c is much larger than k_2^c, then the contact surface will approach the intersection surface of Γ_1, i.e., the surface of Γ_1 behaves rigidly during the contact, whereas the surface of Γ_2 is elastic. We choose $k_i^c = E_i$, where E_i is Young's modulus of a material and thus determines the object's stiffness (see Section 5.3). Hence, the surface of elastic objects (low E) adapts to stiff objects (high E).

To get an initial sampling of $\mathcal{S}_c$, the colliding surfels are moved onto $\mathcal{S}_c$, see Figure 5.18 (b). The sampling points on $\mathcal{S}_c$ are called *contact nodes*. A contact node c consists of two surfels with identical positions but opposing normals, similar to crack nodes (Section 5.6.4). For finding the position $\mathbf{c} \in \mathcal{S}_c$ of a surfel $s_1 \in \mathcal{S}_1$ such that $F(\mathbf{c}) = 0$, the Newton-Raphson method could be used (Section 2.4.2). However, this would require computing $\nabla F(\mathbf{s}_1)$, e.g., using finite differences. Furthermore, this method might not converge to a solution. We therefore reduce the problem of finding the root to a 1D problem by projecting $\mathbf{s}_1$ onto the implicit surface representation Ψ_2 of Γ_2 using the (not necessarily orthogonal) projection operator $\psi_2(\mathbf{x})$ described in Section 2.5.1. The point $\mathbf{c}$ with $F(\mathbf{c}) = 0$ lies on the line between $\mathbf{s}_1$ and $\psi_2(\mathbf{s}_1)$, where $\mathbf{c}$ can be found iteratively using Brent's method as described in Section 2.4.2. If the boundary of the intersecting volume is not convex, it might happen that the line between $\mathbf{s}_1$ and $\psi_2(\mathbf{s}_1)$ is not inside the intersection volume Ω_c. Thus, a new contact node is created only if $\mathbf{c} \in \Omega_c$.

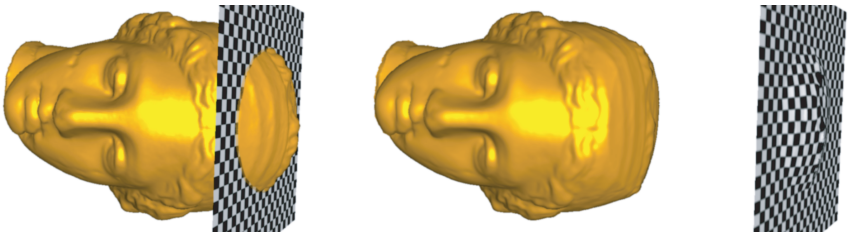


Figure 5.19: Left: collision of Igea with plane without surface contact handling. Middle and right: surface response of Igea and plane, respectively.

To get an even and hole free sampling of the contact surface, a resampling scheme is applied consisting of relaxation and resampling steps similarly to the one in Section 6.6.3. A relaxation force, derived from a *repulsion potential*, locally distributes the surfels evenly. After each repulsion step, the contact nodes are projected onto $\mathcal{S}_c$. Afterwards, contact nodes are inserted or deleted depending on the number of neighboring nodes within a certain distance, see Section 6.6.3 for details. This series of relaxation, projection and resampling steps is repeated until the displacement of nodes is below an error threshold and no resampling takes place anymore, resulting in a fully covered and locally uniformly sampled surface. The combination of both relaxation and resampling results in an efficient algorithm for covering a certain area, as was already stated by Witkin and Heckbert [WH94].

For visual accuracy, the normal vectors are recomputed using principal component analysis (PCA), i.e., the eigenvectors of the covariance matrix of the local neighborhood are computed, where the eigenvector corresponding to the smallest eigenvalue gives the normal direction [HDD+92].

5.8.4 Collision Response

We have shown above how a contact surface can be computed from the colliding surfels and how it is sampled with contact nodes. The information gained during the creation of the contact surface is used to apply a penalty method that separates intersecting objects, i.e., forces are computed that act against the penetration. While penalty methods are efficient to compute and yield stable animations, they do not prevent objects from penetrating. However, since we have already handled penetration by deforming the surface, a simple penalty method is sufficient as collision response model for the volume.

In the following subsections an approach is described that computes a penalty and friction force for each contact node. These forces are then distributed to the particles of the colliding objects. We consider two colliding objects Γ_1 and Γ_2 and derive the forces for Γ_1. The forces for Γ_2 are computed analogously.

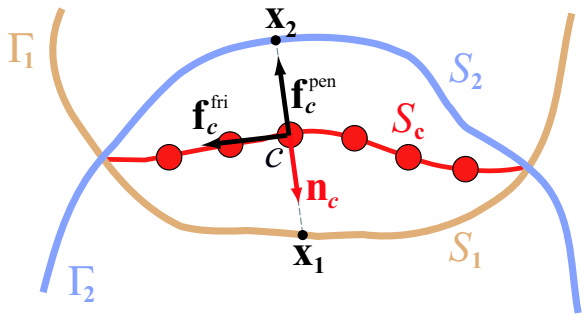


Figure 5.20: Penalty and friction force computation for a contact node c on the contact surface $\mathcal{S}_c$ between two objects Γ_1 and Γ_2 with surface $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. The forces depend linearly on the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$.

Penalty Force Computation

A penalty force is defined for each contact node such that two intersecting objects are separated along the penetration direction. A reasonable approximation of this direction at a contact node c in case of small penetrations is to take the normal vector $\mathbf{n}_c$ at c, see Figure 5.20.

By shooting a ray from $\mathbf{c}$ along $\mathbf{n}_c$ and $-\mathbf{n}_c$ and intersect it with $\mathcal{S}_1$ and $\mathcal{S}_2$ we get the points $\mathbf{x}_1 \in \mathcal{S}_1$ and $\mathbf{x}_2 \in \mathcal{S}_2$, respectively (if the surface is in the middle, i.e., $E_1 = E_2$, then it is sufficient to compute only $\mathbf{x}_1$, mirroring it at $\mathbf{c}$ gives $\mathbf{x}_2$). We then set up a force-displacement relationship between $\mathbf{x}_1$ and $\mathbf{x}_2$. We choose a linear elastic model depending on the penetration direction $\mathbf{d}_c = \mathbf{x}_1 - \mathbf{x}_2$, i.e., a linear spring of rest length $\|\mathbf{d}_c\|/2$ is attached between $\mathbf{x}_1$ and $\mathbf{x}_2$ resulting in the following force

$$\mathbf{f}_c^{\text{pen}} = -\frac{1}{2}(k^{\text{pen}}\mathbf{d}_c + k^{\text{damp}}\dot{\mathbf{d}}_c), \tag{5.38}$$

where k^{pen} is the spring stiffness. This stiffness is a parameter of our collision response model. It controls the trade-off between quality, i.e., small penetrations, and the stability of the simulation. The stability problems in connection with stiff springs is reduced by adding damping. The time derivative of the penetration direction $\dot{\mathbf{d}}_c$ at $\mathbf{c}$ is equal to the relative velocity of Γ_1 and Γ_2 at $\mathbf{c}$ in direction $\mathbf{d}_c$, i.e.,

$$\dot{\mathbf{d}}_c = \mathbf{v}_c^{\text{per}} = \frac{(\mathbf{v}_c^{\text{rel}} \cdot \mathbf{d}_c)\mathbf{d}_c}{\|\mathbf{d}_c\|^2}. \tag{5.39}$$

The relative velocity is computed as

$$\mathbf{v}_c^{\text{rel}} = \mathbf{v}_{\Gamma_1}(\mathbf{c}) - \mathbf{v}_{\Gamma_2}(\mathbf{c}), \tag{5.40}$$

where the velocity $\mathbf{v}_{\Gamma_i}(\mathbf{c})$ of object Γ_i at $\mathbf{c}$ is computed using a normalized SPH approximation of the velocity of particles p_j within the range h_c:

$$\mathbf{v}_{\Gamma_i}(\mathbf{c}) = \frac{\sum_{p_j \in \Omega_i} \omega(\mathbf{c} - \mathbf{p}_j, h_c) V_j \mathbf{v}_j}{\sum_{p_j \in \Omega_i} \omega(\mathbf{c} - \mathbf{p}_j, h_c) V_j}, \tag{5.41}$$

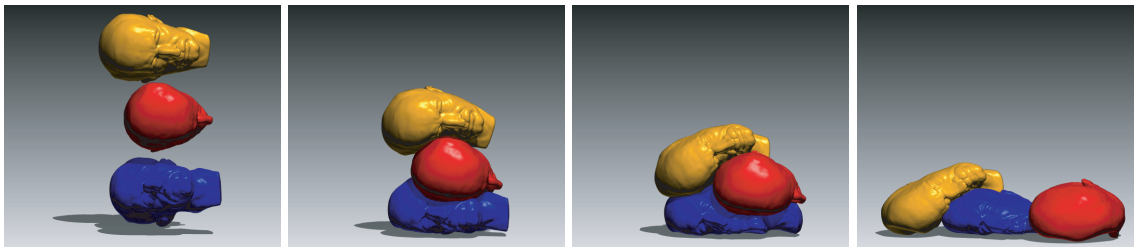


Figure 5.21: Plastic Max Planck models building a pile and falling apart again.

where we use the kernel ω^{poly} as a weighting function (see Equation (4.9)).

Friction Force Computation

To prevent two objects from freely sliding relative to one another, we apply a Coulomb friction force as proposed in [MC95] to each contact node c

$$\mathbf{f}_c^{\text{fri}} = -\mu^{\text{fri}} \|\mathbf{f}_c^{\text{pen}}\| \frac{\mathbf{v}_c^{\text{tan}}}{\|\mathbf{v}_c^{\text{tan}}\|}, \tag{5.42}$$

where μ^{fri} is the coefficient of friction and $\mathbf{v}_c^{\text{tan}}$ the velocity vector tangential to the penetration direction $\mathbf{d}_c$

$$\mathbf{v}_c^{\text{tan}} = \mathbf{v}_c^{\text{rel}} - \mathbf{v}_c^{\text{per}}, \tag{5.43}$$

where $\mathbf{v}_c^{\text{rel}}$ is the relative velocity and $\mathbf{v}_c^{\text{per}}$ the velocity in direction of the penalty force as defined in the previous section.

Force Distribution

The total collision response force $\mathbf{f}_c^{\text{surface}}$ of a contact node $c \in \mathcal{S}_c$ is the sum of the forces defined in Equations (5.38) and (5.42)

$$\mathbf{f}_c^{\text{surface}} = \rho_c(\mathbf{f}_c^{\text{pen}} + \mathbf{f}_c^{\text{fri}}), \tag{5.44}$$

where ρ_c denotes the local contact node density at c. This scales the force such that it is independent of the contact surface sampling. We use a simple estimation for ρ_c as suggested by Pauly [Pau03]

$$\rho_c = \frac{k}{\pi d_c^2}, \tag{5.45}$$

where d_c is the distance to the farthest of the k nearest neighbor nodes of c.

So far the computed contact forces $\mathbf{f}_c^{\text{surface}}$ are defined per contact node. To influence the object's dynamics, these forces need to be distributed to the particles.

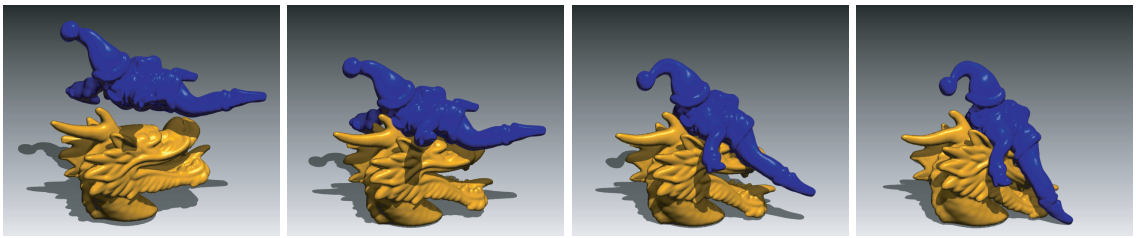


Figure 5.22: Santa Claus riding the dragon.

A contact node c exerts a force $\mathbf{f}_{i,c}^{\text{contact}}$ onto a particle $\mathbf{p}_i \in \Omega_1$ within its support radius h_c

$$\mathbf{f}_{i,c}^{\text{contact}} = \frac{\omega(\mathbf{p}_i - \mathbf{c}, h_c)\mathbf{f}_c^{\text{surface}}}{\sum_j \omega(\mathbf{p}_j - \mathbf{c}, h_c)}, \tag{5.46}$$

where ω is a weighting kernel (we use again ω^{poly}). The force for particles $\mathbf{p}_i \in \Omega_2$ is computed analogously, but with the negative contact force $-\mathbf{f}_c^{\text{surface}}$.

The total force acting on a particle p_i is then

$$\mathbf{f}_i^{\text{contact}} = \sum_{c \in \mathcal{S}_c} \mathbf{f}_{i,c}^{\text{contact}}. \tag{5.47}$$

This force will act on p_i like an external force at the next time step. Note that the sum of all contact node collision forces is equal to the sum of all particle forces acting on an object, i.e., $\sum_c \mathbf{f}_c^{\text{surface}} = \sum_i \mathbf{f}_i^{\text{contact}}$. Because a contact node always exerts the same force onto both objects, the total sum of all exerted forces is zero. The forces therefore conserve linear and angular momentum of the system.

5.8.5 Contact Handling Pipeline

We will now summarize the different steps needed for contact handling between an arbitrary number of objects. After particles and surfels are animated, collision handling for each pair of objects is performed. A bounding box is computed for both objects and the surfels in the intersecting box are checked for collision as described in Section 5.8.2. The colliding surfels are moved to the contact surface $\mathcal{S}_c$. Afterwards, $\mathcal{S}_c$ is resampled and for visual accuracy the normals are recomputed using PCA (Section 5.8.3). The penalty forces are computed for each contact surfel on $\mathcal{S}_c$ and then distributed to the neighboring particles (Section 5.8.4). Finally, the new surface $\mathcal{S}_i^{\text{new}}$ consists of the non-colliding surfels unified with one set of surfels of $\mathcal{S}_c$. Note that after having computed the penalty forces, the new collision free surface $\mathcal{S}_i^{\text{new}}$ is used only for rendering. The next time step operates on the original surface $\mathcal{S}_i$. However, $\mathcal{S}_i^{\text{new}}$ is also used for contact handling with further objects. Contact handling depends thus on the order of the object pairs. However, in practice there is only little difference.

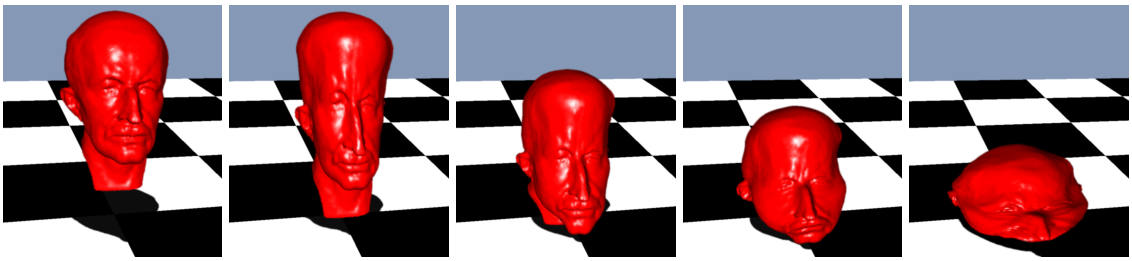


Figure 5.23: A soft elastic Max Planck model is initially held by the cranium, then melted and dropped to the ground plane: 200 particles, 10k surfels, 28 frames/s

5.9 Results & Discussion

With the algorithms described above we developed a framework for the simulation of deformable elasto-plastic objects including fracturing and contact handling. In this section we will discuss the achieved results and limitations of our approaches. For all timings we used a standard Pentium IV 3 GHz PC and a NVidia GeForce FX 5900 GPU.

Our framework allows to change physical properties during run-time, as illustraed in Figure 5.5, where we let an elastic model with $E = 0.5 \cdot 10^6 \mathrm{N/m^2}$ bounce off the ground plane. The model exhibits realistic elastic behavior. Shortly before hitting the ground a second time, we switch to a plastic material, resulting in an irreversible dent. Afterwards, we switch back to a stiff elastic material with $E = 0.5 \cdot 10^7 \mathrm{N/m^2}$. The model has taken considerable damage, but the surface is still skinned correctly. A real-time melting animation with an adaptively sampled surface is shown in Figure 5.23, where the model exhibits realistic elastic, plastic, melting and flowing effects. For elastic deformations we achieve real-time animations with about 30 frames/s for models that are sampled with approximately 200 particles and 10k surfels, where OpenGL is used for rendering.

A complicated and highly detailed octopus model (Figure 5.24) with 2.7k particles and 465k surfels shows the stability and efficiency of our method. In Figure 5.2 we demonstrate the effect of Poisson's ratio ν for volume conservation. For the image in the middle we set ν to zero. When the model is pulled vertically, its width does not change and its volume is not conserved. In contrast, with a ratio of 0.49 the width adjusts to the stretch thereby approximately conserving the volume of the object (right).

Figure 5.7 shows brittle fracture of a stiff elastic object, computed at an average of 22 seconds per frame. The initial model is sampled with 4.3k particles and 249k surfels. During fracturing the number of particles increases to 6.5k, while 61k additional surfels have been created to define the new fracture surfaces. Compared to FEM-based approaches that use the faces of simulation elements to define the object surface, we achieve a significantly higher level of surface detail without requiring a proportionally large number of particles. This decou-

Figure 5.24: Animating a highly detailed octopus model: 2.7k particles, 465k surfels, 2s/frame

pling of the simulation domain from the representation of the boundary surface leads to increased performance and provides essential control in computer animation, where visual quality is typically favored over physical accuracy. On the other end of the spectrum of material properties that our method can handle is the example of Figure 5.8. The highly plastic bubble gum is deformed beyond recognition before splitting along a complex fracture surface. The initial sampling resolution is 2.2k particles and 134k surfels, increasing to 3.3k and 144k, respectively, in the final model. Average simulation time is 2.4 seconds per frame. Figure 5.14 shows how we can explicitly control fracture, using a combination of crack painting, propagation history, and adaptive fracture thresholds. Initially, the model is sampled with 4.6k particles and 49k surfels, fracturing increases the number of particles to 5.8k and the number of surfels to 72k, with an average of 6 seconds per frame. The images of the animation sequences shown in Figures 5.7, 5.8 and 5.14 were created using the open-source renderer POV-Ray (`http://www.povray.org`), which we extended to handle ray intersections with surfels as described in [AA03a, AA04a, AKP+05]. As mentioned above, we avoid an explicit representation of the intersection curve of two adjacent surface sheets by deferring the surface-surface intersection problem to the rendering stage, where it can be solved efficiently. We adapt the CSG rendering technique for point-sampled surfaces proposed by Wicke et al. [WTG04]. During fracturing, we maintain a list of all intersecting surface sheets that form a sharp crease. With this minimal topological structure all surface-surface intersections can be resolved by the renderer during ray-casting (see Figure 5.10).

Note that all fractured models have been resampled substantially during the simulation to match the increased geometric and topological complexity after fracturing. The simplicity of this dynamic resampling of the simulation domain highlights one of the main benefits of meshless methods for physics-based animation. A similar argument holds for our surface sampling method. Instead of maintaining a consistent surface mesh and dynamically cutting and remeshing during simulation, our sampling scheme simply inserts and moves surfels during crack propagation.

An example for contact handling is shown in Figure 5.21, where three plastic

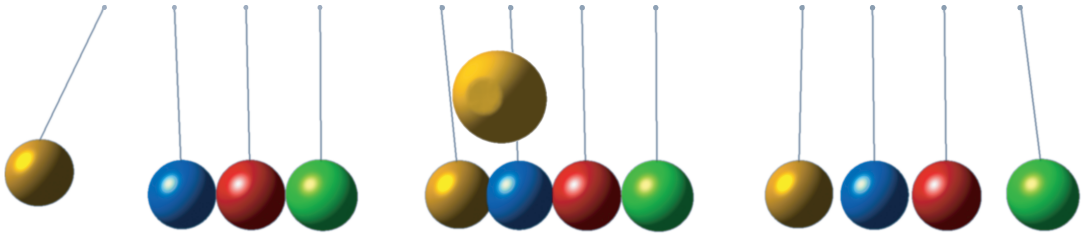


Figure 5.25: Newton's Cradle with stiff elastic spheres. Middle: Contact surface of left sphere.

Max Planck busts fall onto each other building a pile, fall apart again and finally come to rest. Stiff elastic spheres are used to simulate Newton's Cradle, see Figure 5.25. In the middle picture the contact surface of the first sphere is shown. The Santa Claus riding the dragon, shown in Figure 5.22, is an example for contact handling between two highly complex models. To prevent that the mouth of the dragon self-intersects we store all neighbors within the influence distance. If two particles come close that were not in the influence region before, a simple penalty force is applied to the two particles.

In the last example shown in Figure 6.7, a highly plastic object first stretches and then fractures (see Section 6.7 for details). The two separated parts are detected and converted into two separated objects. Afterwards, contact handling between the two objects is performed as described above.

In Figure 5.26, we show the time measurements of two colliding Max Planck busts with each 10k surfels and 390 particles. We made the busts elastic such that the physical animation performs in about constant time. The red line shows the total time needed for the animation including contact handling and rendering. The time needed for contact handling is shown in blue, and the number of colliding surfels is shown in orange. There are many factors influencing the performance: For large intersections that yield contact surfaces with many contact nodes, distributing the node forces to the particles is the most expensive step. This can be done in $O(c \cdot q)$, where c is the number of contact nodes and q is the average number of particles within the support radius h_c of a contact node, where we assume that range queries can be done in constant time using a hash data structure (see Section 7.1.1). The time complexity for computing the colliding surfels depends linearly on the number of surfels in the intersected bounding box. For computing the contact surface, the penetration depth is crucial for the performance. Deeper penetration usually means that more iterations are needed to project surfels onto the contact surface and to resample it.

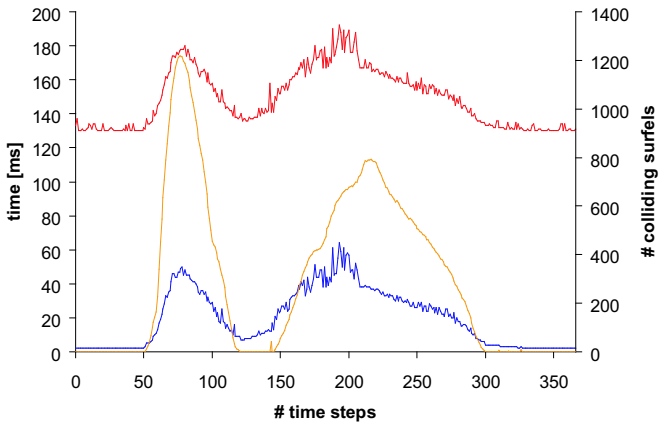


Figure 5.26: Performance measurement for two colliding Max Planck models with each 10k surfels and 390 particles. Red: total animation time. Blue: contact handling time. Orange: number of colliding surfels.

5.9.1 Limitations

The presented framework has several limitations (some are discussed as future work in the next section).

The used elasticity model assumes a *Hookean* material, i.e., materials with a linear stress-strain relationship. However, this does not restrict us to isotropic materials. Anisotropic materials can be simulated by modifying the constitutive matrix $\mathbf{C}$ described in Section 5.3.

The moving least-squares approximation method used for first-order accurate approximation of the gradient of the displacement field (Section 5.4.1) only works well if each particle has at least three neighbors at non-degenerated locations. The approach thus only works for volumes, not for 2D layers or 1D strings of particles. Note that a minimum number of neighbors can be enforced using the resampling algorithm described in Section 5.7. However, if, for instance, during fracturing many small fragments are created, the number of particles increases drastically which slows-down the computation significantly. Therefore, we resort to modeling small pieces of material as rigid bodies, assuming that the internal deformations are negligible. Fortunately, large numbers of very small fragments are mainly created by stiff objects that experience brittle fracture, where such an approximation is reasonable.

In the model described in this chapter, the rest shape is fixed or transformed rigidly. Thus, the neighborhood of the particles does not change which prevents from recomputing the moment matrices and weights after each time step. However, changes in topology, which are not modeled explicitly as was done for fracturing, cannot be handled. This restriction will be lifted in the next chapter by using a deformable rest shape, see Section 6.4.1.

Extremely stiff objects require very small time steps to obtain an accurate distribution of stress within the material. This problem can be alleviated using implicit or semi-implicit integration schemes. A derivation of the stiffness matrix and the implicit dynamic equations are given in [MKN+04].

The presented collision handling algorithm uses penalty forces to separate penetrating objects. The computation of these forces depends on an accurate estimation of the penetration direction. The algorithm is therefore especially susceptible to fail for deep penetrations, where the penalty force might point to other directions than the actual deformation direction. Also the computation of the contact surface can be very costly for deep penetrations. Furthermore, the algorithm is not capable to detect and resolve self-intersections.

5.10 Extensions & Future Work

The framework has been extended in several ways and opens up various research directions.

As described in Section 5.4.3, the rest shape is transformed rigidly to reduce numerical problems and ghost forces. This works fine as long as the whole object undergoes more or less rigid transformations. However, if two or more parts of an object are transformed independently, these problems can still occur. A possible solution is to divide the set of particles into overlapping clusters, similar to [MHTG05]. The shape matching is then done per cluster, and the obtained positions blended with overlapping clusters. If we increase the number of clusters so that at the end every particle with its surrounding neighbors build a cluster, the rigid transformations are extracted per particle. However, then it will take longer for a deformed object to turn back to its rest shape. Furthermore, it increases the needed computation time per particle significantly.

Our fracturing algorithms can also be used for cutting, as demonstrated by Steinemann et al. [SOG06]. To make it suitable for interactive simulations, they construct a visibility graph that stores proximity information. The visibility graph is updated dynamically when a crack propagates.

In Section 4.4 an approach has been presented for fluid-rigid body interaction. The rigid body is sampled with particles and then treated as a rigid fluid, i.e., forces between fluid and rigid body particles are computed the same way as for multiphase fluids and then integrated onto the rigid object to achieve rigid body motion. We can use the same approach for simulating the interaction between fluid and deformable objects. The forces applied from the fluid onto the deformable object particles are added as external forces. Experiments show that this works very stably. As with rigid bodies, the sampling density of the deformable object should be similar to the sampling of the fluid. Unlike with rigid bodies where it is sufficient to only sample the object with particles within the distance of the support radius, the whole volume of the deforming objects needs to be sampled.

However, to reduce the number of solid particles in the interior of the object, the multiresolution approach described in Section 4.5 can be used.

The presented contact handling algorithm can only detect collisions between separated objects or object parts. Efficient detection of self-collisions is a difficult problem, especially in the case of point-sampled surfaces due to the missing connectivity. Exploiting spatial and temporal coherence for contact handling would make our algorithm more efficient. In [HTK+04] we propose a method to compute a consistent penetration depth and direction even in case of large penetrations. Although this approach requires a volumetric mesh, it could be adapted to particles.

In [AKP+05] we exploit the surface animation scheme presented in Section 5.5 to accelerate building a bounding sphere hierarchy for efficient raytracing of deforming point-sampled objects. Due to the decoupling of the (high-resolution) surface and the (coarse) volumetric representation, a reduced deformation model similar to [JP04] is used, where the displacement of the surface can be expressed with significantly fewer parameters than the number of degrees of freedom of the surface itself. Furthermore, a novel technique for rendering sharp edges and corners in point-sampled models is presented. The update of the bounding sphere hierarchy could also be used for accelerating the collision detection described in Section 5.8.2.

In the next chapter we will describe how to combine fluids and deformable objects to simulate viscoelastic materials as well as phase-transitions, i.e., melting and freezing. This also requires an extension of the surface animation such that implicit changes in topology can be handled and a smooth transition from a detailed solid surface to a smooth fluid surface can be achieved.

5.11 Summary

With the meshless framework described above, deformable objects with material properties ranging from stiff elastic to highly plastic can be simulated at interactive rates for moderate complexity. The forces are derived from continuum mechanics and computed locally based on the neighborhood of a particle. Physical parameters can be modified at run-time, resulting in visually plausible and interesting effects such as melting and solidifying as also described in the next chapter.

Extending this framework for fracturing is straightforward and shows several advantages compared to FEM simulation. Instead of maintaining a consistent volumetric mesh using continuous cutting and restructuring of finite elements, the shape functions of the particles are adjusted dynamically based on simple visibility constraints.

The space discretization is continuously adapted using insertions of new particles. The simplicity of this dynamic resampling of the simulation domain highlights one of the main benefits of meshless Lagrangian methods for physics-based animation. Due to minimal consistency constraints between neighboring parti-

cles, dynamic resampling is efficient and easy to implement, as compared to the far more involved remeshing methods used in FEM simulations.

A point-based representation is built for the boundary surface, which allows efficient dynamic sampling of fracture and contact surfaces as well as surface refinement to maintain a high quality surface also for large deformations. Furthermore, it facilitates explicit control of the object topology (see also Section 6.6). The decoupling of the surface from the volumetric representation enables efficient animations of highly detailed surfaces.

This decoupling is also exploited for resolving object-object collisions. By using a high resolution surface for contact handling and a low resolution representation for the deformation, the animation is both stable and efficient. Due to the smoothing effect of the distribution of the surface forces to the particles, oscillations for resting contacts are avoided.

A general limitation of the meshless approach is that even very small fragments must be sampled sufficiently dense in order to obtain a stable evaluation of the shape functions. This inflates the number of particles when an object is fractured excessively, which slows down the computations.

Chapter 6

Solid-Fluid Simulation

In this chapter we want to extend and combine the fluid and deformable solids animation frameworks described in Chapter 4 and Chapter 5 such that we can simulate also materials that are in-between fluids and solids, so-called viscoelastic materials. We briefly discuss how to derive a common equation for both solids and fluids from a Lagrangian viewpoint. Using our particle-based approach, we are able to employ a unified method to animate both solids and fluids as well as viscoelastic materials and melting and freezing. Central to our framework is a hybrid implicit-explicit surface generation approach that is capable of representing fine surface detail as well as handling topological changes at interactive rates for moderately complex objects. We thus extend the point-based surface model described in the last chapter such that the surface adapts to the new position of the particles by minimizing a potential energy subject to geometric constraints.

Our physics model is based on Newtonian mechanics and the combination of fluid and elastic solid stresses (Section 6.3). Arbitrary plastic deformations can be achieved by adapting the rest shape to the deformation (Section 6.4). By introducing temperature diffusion and a simple model for coupling temperature with physical parameters, melting and freezing effects can be simulated (Section 6.5). To cope with these effects, potential fields are defined due to which the surface deforms such that it adapts to the physics representation (Section 6.6). Our surface model combines the advantages of explicit and implicit representations in that it can explicitly represent high detailed surfaces for solids, but continuously changes to an implicit representation while melting to a fluid with a smooth surface (Section 6.6.6). We illustrate our algorithm on a variety of examples ranging from stiff elastic and elasto-plastic materials to fluids with variable viscosity, as well as melting objects and freezing fluids (Section 6.7).

6.1 Introduction

Simulation of special effects, such as melting and freezing or the animation of viscoelastic materials, play an important role in computer graphics applications such as feature films and games. A prominent example is the terminator sequence from the well-known feature film, where the metallic terminator is shattered, after which the individual pieces melt and fuse, before retaining the old shape and freezing to a solid. Filming these effects is often not possible, be it because it is too costly or too complex. Simulation on the computer has the advantage that it is comparably cheap, and even materials can be animated that do not necessarily exist. However, even if the simulated materials are artificial, the animations still have to look physically plausible.

The main difference between solids and fluids is that solids have restoring forces due to stresses, whereas an ideal Newtonian fluid stores no deformation energy. However, many materials cannot be classified clearly as solid or fluid. Solids often start to flow plastically when a high continuous stress is applied. On the other hand, some (so-called non-Newtonian) fluids can withstand small shearing stress, see also Section 2.1.5. To simulate all kinds of these so-called *viscoelastic* materials, a method has to be able to handle the full range from stiffelastic solids to elasto-plastic materials to fluids. In Chapters 4 and 5 we applied the meshless Smoothed Particle Hydrodynamics (SPH) method to simulate fluids and elasto-plastic solids, respectively. In this chapter we will combine these approaches such that arbitrary viscoelastic materials can be simulated. Furthermore, we will show how we can continuously change between the material properties depending on a temperature scalar value to animate materials that melt and freeze.

Meshless particle-based methods require the definition or extraction of an implicit or explicit surface, for instance, for rendering and contact handling. In the context of a unifying framework for solid-fluid animations the surface must be able to fulfill various requirements. Solid surfaces are often very detailed, whereas fluid surfaces are smooth due to surface tension. For phase transitions from solids to fluids the detail should disappear, whereas from fluids to solids the existing detail has to be preserved. Additionally, the surface should be temporally smooth, i.e., temporal aliasing such as popping artifacts have to be avoided. Finally, topological changes such as splashes must be handled by the surface. We will show how to fulfill these requirements by dynamically maintaining a point-sampled surface wrapped around the particles.

6.2 Related Work

In this section we give a short overview of work that has been done in physics-based animation of melting and solidifying of objects and the simulation of viscoelastic fluids. Furthermore, we will discuss research in surface extraction and

animation for simulated deforming objects and fluids.

6.2.1 Melting Objects and Viscoelastic Fluids

Terzopoulos et al. extended their work on physics-based animation of deformable objects [TPBF87, TF88] for heating and melting [TPF89]. In their mass-spring system, they achieved a (local) phase transition from solids to fluids by simply varying the spring constant and finally removing a spring. Tonnesen [Ton91, Ton98] couple particles based on potential energies (see Section 3.1.1 for details). Applying the heat equation to this particle system changes the potential energies, thus, soft objects start to melt. Carlson et al. [CMVT02] used an Eulerian grid-based fluid simulation method for melting, flowing and solidifying of objects. Their method is capable of modelling different materials, ranging from rigid solids to fluids, by varying the viscosity. Stora et al. [SAC+99] simulated the flow of lava using SPH by coupling viscosity with temperature. Viscoelastic and plastic materials are achieved by Clavet et al. by combining the SPH method with a mass-spring system [CBP05]. Wicke et al. [WHP+06] discard explicit connectivity information completely. Instead, they use the initial regular particle sampling as a rest state in combination with shape matching techniques.

Goktekin et al. [GBO04] added elastic terms to the Navier-Stokes equations which they solve using Eulerian methods, thus obtaining viscoelastic fluids that can model a variety of materials such as clay and pudding. Our work is similar to theirs in that we also combine fluid and solid characteristics. Solving an equation that combines solid mechanics and the Navier-Stokes equations allows us to animate materials from stiff elastic and elasto-plastic objects to fluids with low viscosity. Losasso et al. [LIGF06] use a FEM simulation for deformable objects and a Eulerian solver for fluids. A level set representation for the surface is used to couple the two methods, thus enabling melting and burning of solids. However, the representation of fine surface detail is difficult using level sets.

6.2.2 Surface Extraction and Animation

State of the art methods in Eulerian fluid simulation use level sets, introduced by Osher and Sethian [OS88], to render the fluid [FF01]. Level sets start with an implicit function that is evolved over time using a velocity field. This allows temporally smooth surface animation. However, level set evolution can suffer from severe volume loss, especially near detailed features such as splashes. As a solution, Enright et al. propose to combine level sets with surface particles [EMF02, ELF05a]. Volume loss is avoided by Bargteil et al. [BGOS06] who compute the advected grid distance values to an explicit mesh, thus obtaining exact values. This comes at the price of lower computational speed.

Desbrun and Canni model soft inelastic objects that split and merge by coating a set of skeletons using an implicit representation [DC95]. They extend their

work by introducing active implicit surfaces [DC98], which move according to a velocity field. The velocity field is chosen such that the surface is attracted to a geometric coating, but other terms such as surface tension and volume conservation are also applied.

The techniques discussed above animate the surface by solving a PDE on a grid, followed by isosurface extraction for rendering. Witkin and Heckbert define constraints to keep surface particles on a moving implicit surface [WH94]. Surface particles adaptively sample the surface using a splitting and merging scheme. Szeliski and Tonnesen introduced oriented particles for surface modeling [ST92]. Additional to long-range attraction and short-range repulsion forces, they define potentials that favor locally planar or locally spherical arrangements. Their particle system can handle geometric surfaces with arbitrary topology. Similar to them, we use surfels as oriented particles. Our explicit surface representation deforms by internal and external forces and geometric constraints, similar to active implicit surfaces [DC98]. We derive the forces by minimizing a potential energy term that depends on both the surface and the physical particles. Besides gaining efficiency in computation and memory, we can exploit all advantages of explicit surfaces such as the simple representation of fine surface details.

6.3 Physics Model

The difference in the physics model between fluids and elastic solids is that solids have a rest shape defined and thus elastic restoring forces due to internal stress. In the next section we will briefly discuss the equations for fluids and solids, and derive a unified momentum equation that enables also the simulation of viscoelastic materials.

6.3.1 Governing Equations

As described in Section 2.1 (see also [MG04]), we can write the equations for an elastic solid as

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \sigma^s(\mathbf{u}) + \tilde{\mathbf{f}}^{\text{ext}}, \tag{6.1}$$

and for an incompressible Newtonian fluid as

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \sigma^f(\mathbf{v}) + \tilde{\mathbf{f}}^{\text{ext}}. \tag{6.2}$$

Both Equations (6.1) and (6.2) describe the change in momentum, which is equal to internal force density fields due to stresses and body force density vector fields $\tilde{\mathbf{f}}^{\text{ext}}$, where ρ denotes the density, $\mathbf{u}$ the displacement from the material coordinates $\mathbf{m}$, $\mathbf{v}$ the velocity and σ the stress tensor. $\frac{D\mathbf{v}}{Dt}$ is the material derivative of the

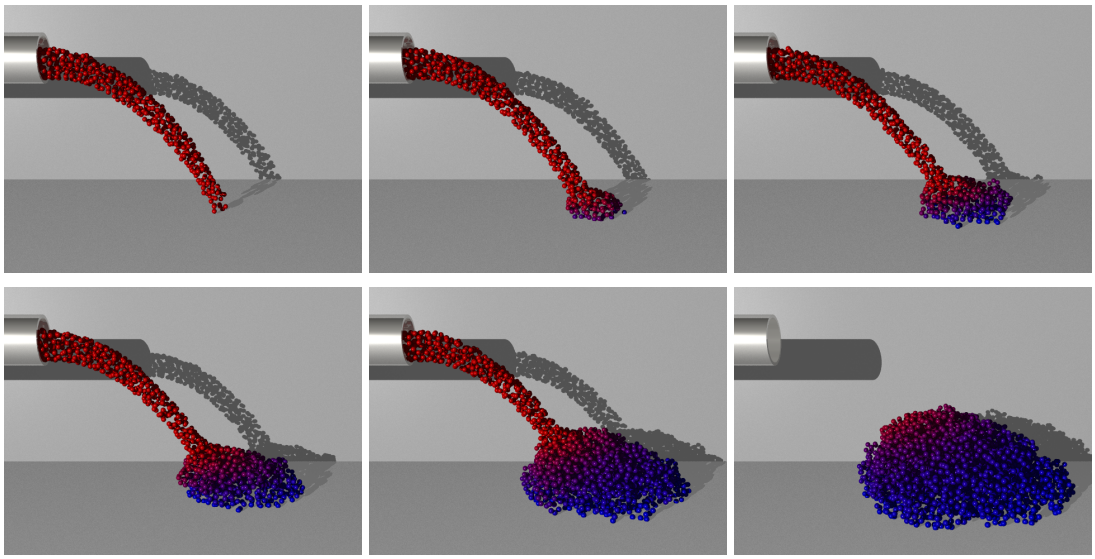


Figure 6.1: Solidifying a fluid due to the contact with the frozen ground.

velocity field. Conservation of mass is represented as

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \mathbf{v} = 0. \tag{6.3}$$

These equations can be simplified when using a particle-based Lagrangian approach. By assigning each particle p_i a constant mass m_i, mass conservation is guaranteed and Equation (6.3) can be omitted. Furthermore, because the particles move with the fluid, the material derivative $\frac{D\mathbf{v}}{Dt}$ of the velocity field is equal to the partial time derivative of the particles' velocity (see Equation (2.5)). Using that $\mathbf{v} = \frac{\partial \mathbf{u}}{\partial t}$, Equations (6.1) and (6.2) can be written in the Lagrangian form as

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \sigma^{\mathrm{s}}(\mathbf{u}) + \tilde{\mathbf{f}}^{\mathrm{ext}}, \tag{6.4}$$

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \sigma^{\mathrm{f}}(\mathbf{v}) + \tilde{\mathbf{f}}^{\mathrm{ext}}. \tag{6.5}$$

The fluid stress tensor σ^{f} is composed of the viscous stress $\sigma^{\mathrm{viscous}}$ and the *isotropic* pressure stress $\sigma^{\mathrm{pressure}}$, i.e., $\sigma^{\mathrm{f}} = \sigma^{\mathrm{viscous}} + \sigma^{\mathrm{pressure}}$. Note that the solid stress σ^{s} is related to shape deformations, whereas the viscous stress $\sigma^{\mathrm{viscous}}$ is due to "sliding forces" that do not change the volume and the isotropic pressure stress $\sigma^{\mathrm{pressure}}$ is due to volume changes. For materials that do not change their volume, we can thus merge Equations (6.4) and (6.5) as follows

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \sigma(\mathbf{u}, \mathbf{v}) + \tilde{\mathbf{f}}^{\mathrm{ext}}, \tag{6.6}$$

where $\sigma(\mathbf{u}, \mathbf{v}) = \sigma^{\mathrm{s}}(\mathbf{u}) + \sigma^{\mathrm{f}}(\mathbf{v})$ is the sum of the elastic, viscous and pressure stress.

6.3.2 Force Computations

For computing the stresses $\sigma^s(\mathbf{u})$ and $\sigma^f(\mathbf{v})$ and the resulting forces, the SPH method described in Sections 4.3 and 5.4 is used. We give here a short summary of the derived forces.

Elastic Force. As described in Section 5.3, the elastic force for a particle p_i can be computed via the strain energy

$$U_i^{\text{strain}} = \frac{1}{2} V_i (\varepsilon_i^s \cdot \sigma_i^s), \tag{6.7}$$

where ε_i^s is the strain and σ_i^s the elastic stress of p_i. We use a linear stress-strain relationship (Hooke's law), i.e., $\sigma_i^s = \mathbf{C}\varepsilon_i^s$. For an isotropic material, the constitutive matrix $\mathbf{C}$ only depends on Young's modulus E and Poisson's ratio ν. The strain is measured using the quadratic Green-Saint-Venant strain tensor

$$\varepsilon_i^s = \frac{1}{2}(\nabla \mathbf{u}_i + \nabla \mathbf{u}_i^T + \nabla \mathbf{u}_i \nabla \mathbf{u}_i^T), \tag{6.8}$$

where $\mathbf{u}_i$ is the displacement of a particle from its rest shape. The elastic force $\mathbf{f}_i^{\text{elastic}}$ of a particle p_i is computed as the directional derivative of the strain energy

$$\mathbf{f}_i^{\text{elastic}} = -\nabla_{\mathbf{u}_i} U_i^{\text{strain}} = -V_i \sigma_i^s \nabla_{\mathbf{u}_i} \varepsilon_i^s, \tag{6.9}$$

where V_i is the volume of p_i, computed as $V_i = m_i/\rho_i$. For modeling plasticity we use the approach described in Section 5.4.4.

Viscosity and Pressure Force. The viscous stress $\sigma_i^{\text{viscous}}$ and the isotropic pressure stress $\sigma_i^{\text{pressure}}$ for a particle p_i are computed as

$$\sigma_i^{\text{viscous}} = \mu_i \nabla \mathbf{v}_i \text{ and} \tag{6.10}$$

$$\sigma_i^{\text{pressure}} = -P\mathbf{I}, \tag{6.11}$$

where μ_i is the viscosity coefficient, P the scalar pressure and $\mathbf{I}$ the identity matrix. The pressure is computed using the constitutive equation given in Equation (4.8). Note that for melting and freezing, μ_i can locally differ and is subject to change. Applying the SPH approximation, we get the following forces

$$\mathbf{f}_i^{\text{pressure}} = -V_i \sum_j V_j \frac{P_i + P_j}{2} \nabla \omega^{\text{spiky}}(\mathbf{r}_{ij}, h) \quad \text{and} \tag{6.12}$$

$$\mathbf{f}_i^{\text{viscosity}} = V_i \sum_j V_j \frac{\mu_i + \mu_j}{2} (\mathbf{v}_j - \mathbf{v}_i) \nabla^2 \omega^{\text{laplace}}(\mathbf{r}_{ij}, h), \tag{6.13}$$

where $\mathbf{r}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ is the distance vector between two particles p_i and p_j and h is the support radius.

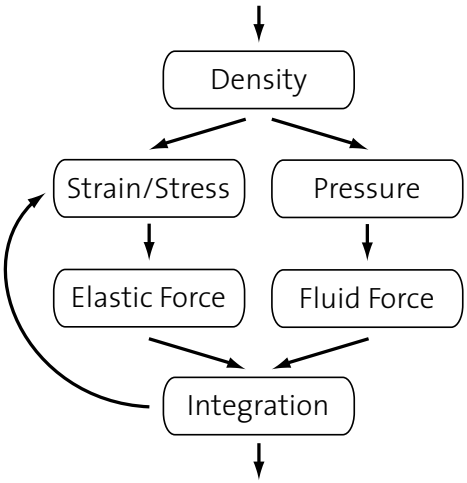


Figure 6.2: Force computation pipeline. The fluid forces are computed only once per time step, whereas the elastic force is computed in every sub time step. This pipeline is part of the animation loop shown in Figure 1.1.

6.4 Particle Animation

Depending on the application, we either sample the volume of an object (see Sections 4.3.1 and 5.7) or use a source that creates a stream of particles. In each iteration, we first compute for each particle p_i its particle neighborhood using either a kd-tree or hash grid as a search data structure (see Section 7.1). The forces are then computed as described in the previous section, yielding the total force

$$\mathbf{f}_i^{\text{total}} = \mathbf{f}_i^{\text{pressure}} + \mathbf{f}_i^{\text{viscosity}} + \mathbf{f}_i^{\text{elastic}} + \mathbf{f}_i^{\text{ext}} \tag{6.14}$$

where $\mathbf{f}_i^{\text{ext}} = m_i\mathbf{g} + \mathbf{f}_i^{\text{contact}}$ is the external body force due to gravity and the collision response force $\mathbf{f}_i^{\text{contact}}$ (see Section 5.8.4). For integration we use the leapfrog scheme (see Section 7.2.1), which showed to be both efficient and stable for our animations.

6.4.1 Deformation of the Rest Shape

All elastic forces are computed relative to a rest shape to counteract derivations of the deformed shape in world coordinates to the rest shape in material coordinates (Section 2.1). To avoid numerical problems, a method has been proposed in Section 5.4.3 to rigidly transform the rest shape such that it optimally matches the shape in world coordinates. However, this does not change the neighborhood of a particle, even if in world coordinates the neighborhood might be completely different. In case of melting, the object can deviate far from its original shape and the topology might change completely. Thus, storing the original neighborhood is not useful anymore. Instead, we suggest to update the reference system by absorbing

the deformation completely while storing the built up strains ε_i^s of a particle p_i in the plastic strain state variable ε_i^p (see Section 5.4.4), i.e.,

$$\varepsilon_i^p \leftarrow \varepsilon_i^p - \varepsilon_i^s, \quad \mathbf{m}_i \leftarrow \mathbf{m}_i + \mathbf{u}_i, \quad \mathbf{u}_i \leftarrow \mathbf{0}, \tag{6.15}$$

where the strain ε_i^s is computed as described in Section 5.4. Note that when we update the rest shape, all information about the original shape is lost and thus the object will not go back to its original shape anymore. Because generally several iterations are needed to bring a deformed mesh back to its original shape, only highly plastic materials can be simulated if the rest shape is updated in every time step. Instead, we update the rest shape only every n-th time step, where $n = 10$ showed to be sufficient for small elastic deformations. It turned out that the required time step for stably integrating the elastic forces is about ten times lower than for integrating the fluid forces (for fluid simulation (Chapter 4) we used a time step $\Delta t = 1$ms and for simulation of deformable objects (Chapter 5) we used $\Delta t = 0.1$ms). The simulation can thus be speeded up by first computing both elastic and fluid forces, but then integrate these forces during ten sub time steps (i.e., with a time step $\Delta t/10$), where the elastic forces are recomputed in every sub time step (cf. Figure 6.2). This results in a significant performance gain because the neighborhood does not need to be recomputed until the rest shape is updated.

6.5 Melting and Solidifying

In our setting a material can be defined by the following main properties: Stiffness (Young's modulus E), compressibility (Poisson's ratio ν), plasticity (k^{yield} and k^{creep}), viscosity (μ) and cohesion for particles at the interface (k^{cohesion}, modeled as pressure jump, see Section 4.4.2). As long as these values are chosen in a (physically) reasonable range, they can be arbitrarily combined. Therefore, also materials that do not exist in the physical world, such as stiff elastic fluids, can be simulated.

For melting and solidifying the material parameters need to change together with the changing aggregation state. To allow local changes, each particle stores its own material parameters. If the material melts from solid to fluid, the stiffness and viscosity decrease, whereas cohesion at the surface and plasticity increase, and vice versa for solidifying.

Note that here we are not interested in the (highly complex) physical accurate modeling of phase transitions. Instead, the user can set the parameters described above for two materials. We assign to each particle p_i a scalar value T_i which is used to interpolate between the two materials. We call T_i the *temperature* of p_i. Assuming a scalar material parameter a is set by the user to $a^{\min}$ and $a^{\max}$ for $T^{\min}$ and $T^{\max}$, a_i is computed using linear interpolation from the current temperature

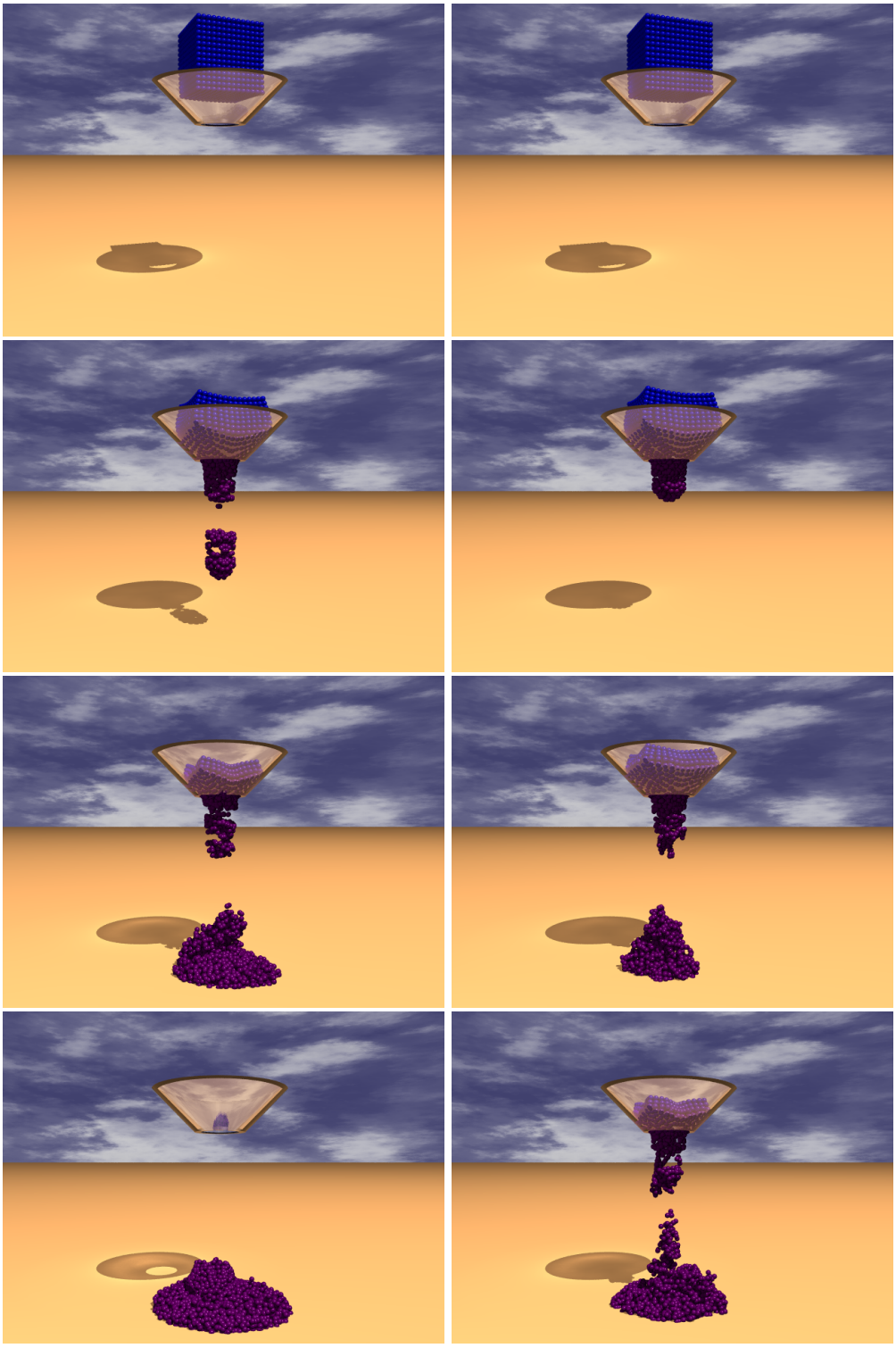


Figure 6.3: Melting two cubes through a funnel with different plasticity constants. The melted cubes freeze on the ground.

T_i

$$a_i = a^{\min} + \frac{T_i - T^{\min}}{T^{\max} - T^{\min}}(a^{\max} - a^{\min}) \tag{6.16}$$

Heat transfer between particles is modeled by solving the heat conduction equation [MHW05]

$$k^{\text{cap}} \frac{dT}{dt} = \frac{1}{\rho}\nabla(\kappa \nabla T), \tag{6.17}$$

where k^{cap} is the heat capacity per unit mass at constant pressure, ρ the density and κ the thermal conductivity coefficient. Assuming that κ is constant and that the heat capacity is constant within the rest volume V_i of a particle $\mathbf{p}_i$ (i.e., $k^{\text{cap}}/m_i = \text{const}$, where m_i is the mass of p_i), we can discretize this equation as follows

$$k^{\text{cap}} \frac{dT_i}{dt} = V_i \kappa \nabla^2 T, \tag{6.18}$$

where ∇^2 is the Laplacian. Solving this equation using SPH yields

$$\frac{dT_i}{dt} = k^{\text{heat}} V_i \sum_j V_j (T_j - T_i) \nabla^2 \omega^{\text{laplace}}(\mathbf{r}_{ij}, h), \tag{6.19}$$

where $k^{\text{heat}} = \kappa / k^{\text{cap}}$ and ω^{laplace} is the same kernel as we used for computing the viscosity forces (see Section 4.3.2). For computing the heat transfer from boundaries to fluid particles we use a model similar to [SAC+99]

$$\frac{dT_i^{\text{ext}}}{dt} = \begin{cases} k_{\text{ext}}^{\text{heat}} V_i (h^2 - d_i^2)\pi (T_{\text{ext}} - T_i), & \text{if } d_i < h \\ 0 & \text{otherwise} \end{cases} \tag{6.20}$$

where d_i is the distance of p_i to the boundary, $(h^2 - d_i^2)\pi$ is the approximation of the contact surface area with the boundary, $k_{\text{ext}}^{\text{heat}} = \kappa_{\text{ext}} / k_{\text{ext}}^{\text{cap}}$, and h is the characteristic smoothing length.

6.6 Surface Animation

In Section 5.5 a surface model has been presented where a point-sampled surface is embedded into the particle representation by advecting the surfels along with the displacement field defined by the particles. While this approach is fast and yields good results in case of deformations, it cannot handle topological changes of the surface, cf. Figure 6.7. In this section we will show how to extend the surface displacement approach such that arbitrary changes of topology defined by the particles can be handled.

To be able to animate and deform the surface, each surfel stores a set of neighboring particles and a set of neighboring surfels (Section 6.6.1). After performing an animation step of the particles, we get an estimation of the new surface by advecting the surfels along with the neighboring particles. Similar to active contour

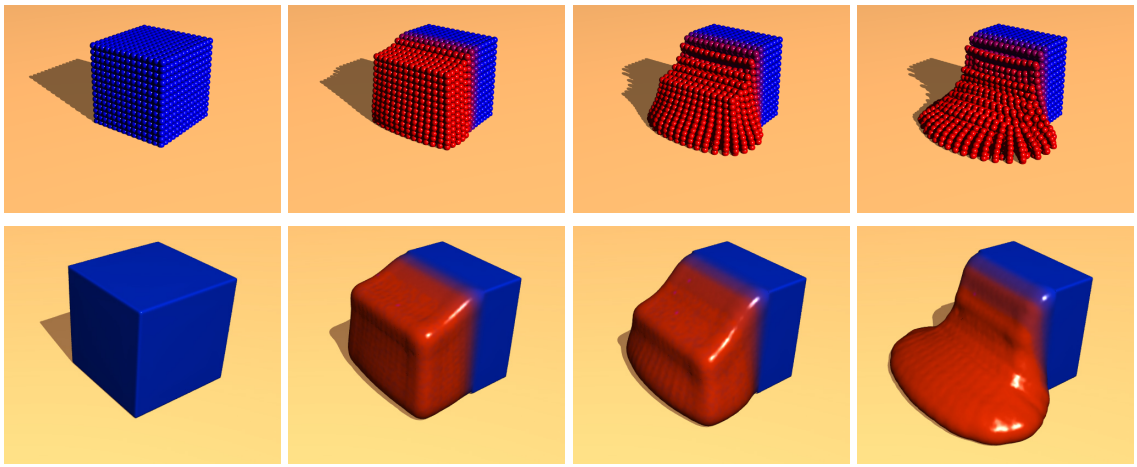


Figure 6.4: Melting of a cube with sharp edges.

models [KWT88, DC98], the surface adapts to the new position of the particles by minimizing a potential energy term while fulfilling geometric constraints (Section 6.6.2). The surface resolution is adapted using a simple resampling scheme that ensures a hole-free surface (Section 6.6.3). Disjoint components of the surface are identified and the particles and surfels are separated accordingly (Section 6.6.5). We detect intersections of separated surfaces and merge the intersecting surface parts (Section 6.6.5). Finally, we show how the surface can be blended smoothly between detailed solid and smooth fluid surfaces (Section 6.6.6).

6.6.1 Surfel Neighborhoods

Each surfel s_i stores a set $\mathcal{P}_i$ of neighboring particles and a set $\mathcal{S}_i$ of neighboring surfels of the same object (Figure 6.5). The neighboring particles are used to estimate the displacement of the surfels after an animation step. From the surfel neighborhood, forces are computed to update the surface as described below.

Particle Neighborhood. The neighboring particles are computed in a pre-animation step. We reuse the search data structure of the particles (Section 6.4) to determine the neighboring particles p_j that are within the (particle) support radius $h_{\mathcal{P}}$ of a surfel s_i with position $\mathbf{s}_i$. Furthermore, a weight $\omega^{\text{poly}}(\mathbf{s}_i - \mathbf{p}_j, h_{\mathcal{P}})$ is stored for each neighbor particle, where the kernel ω^{poly} is used as a smoothly decaying weight function (see Equation (4.9)).

Surfel Neighborhood. Additional to the particle neighborhood, a surfel s_i also stores the neighboring surfels within the (surfel) support radius $h_{\mathcal{S}}$. This neighborhood is recomputed after the surface has been displaced along with the particles. During surface update, the position of the surfels change frequently and therefore this surfel neighborhood needs to be updated quite often. Instead of doing an update of the search data structure and expensive recomputation of the neighbors each time the position of a surfel changes, the following update scheme is used.

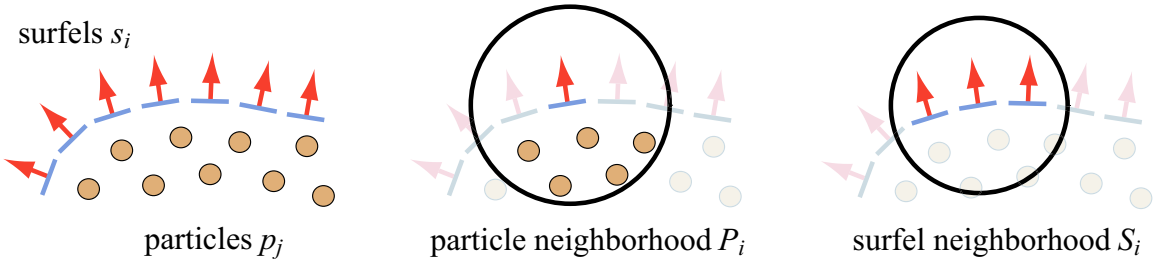


Figure 6.5: Left: the surfels s_i are wrapped around the particles p_j. Middle and right: each surfel s_i stores a particle neighborhood $\mathcal{P}_i$ and a surfel neighborhood $\mathcal{S}_i$.

Assume that a surfel s_i changes its position. Let the new neighborhood be the (initially empty) set $\mathcal{S}_i'$. First, all surfels in $\mathcal{S}_i$ with an Euclidean distance smaller than $h_\mathcal{S}$ are added to $\mathcal{S}_i'$. We then iterate through all neighbors of the neighbors in $\mathcal{S}_i$ and add them to $\mathcal{S}_i'$ if their Euclidean distance is smaller than $h_\mathcal{S}$. By tagging neighbors that were already visited, this update procedure can be performed very fast. Note that not necessarily all neighbors are found, however, as the surfel position does not change significantly during the surface deformation, this update procedure showed to be sufficient. Some of the following algorithms will only use the k nearest neighbors from the neighbor set $\mathcal{S}_i$ (we use $k = 10$). We denote this subset as $\mathcal{S}_i^{\text{sub}}$.

6.6.2 Surface Deformation

The surface animation is divided into two steps: First, the surfels are displaced along with the particles. They are then updated to reflect the new particle positions by minimizing the surface energy, where constraints restrict the possible movements .

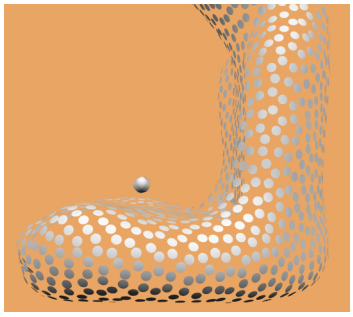
Surface Displacement

In the first step, each surfel s with material coordinates $\mathbf{m}_s$ is displaced using the approach described in Section 5.5.1

$$\mathbf{s} = \mathbf{m}_s + \frac{1}{\sum_j \omega^{\text{poly}}(\mathbf{r}_{s,j}, h_\mathcal{P})} \sum_j \omega^{\text{poly}}(\mathbf{r}_{s,j}, h_\mathcal{P}) \left(\mathbf{u}_j + \nabla \mathbf{u}_j^T(\mathbf{r}_{s,j})\right), \tag{6.21}$$

where $\mathbf{r}_{s,j} = \mathbf{m}_s - \mathbf{m}_j$ is the distance vector between the surfel s and a neighboring particle $\mathbf{m}_j$ in the rest shape, and $\mathbf{s}$ is the position of s in world coordinates. Similarly, the tangent vectors of s are displaced yielding the surfel normal $\mathbf{n}_s$, see Section 5.5 for details.

Surface Update



After advecting the surface along with the particles, it is deformed in normal direction under the action of surface forces, similar to balloons [Coh91]. The forces are derived by minimizing the potential energy of the surface. The potential energy is composed of external potentials that depend on the particles and internal potentials that depend on the surfels. We derive an implicit and an attracting potential such that the energy is minimized when the surfels are attracted to an implicit surface and to the particles, respectively. Minimizing the internal potentials, consisting of the smoothing potential and the repulsion potential, yields a locally smooth and uniformly sampled surface. From the potential energy we derive forces acting on the surfels. The derived forces from the guiding, attracting and smoothing potential act in normal direction, whereas the repulsion force is applied in tangential direction. Note that these forces are defined for deforming the surface only, and thus have no influence on the particles.

Guiding Potential. We define a purely geometric implicit coating of the particles that attracts our explicit surface, similar to Desbrun and Cani [DC98]. Each particle p_j defines a local field function. A potential field is defined by computing the weighted sum of the field functions at an arbitrary position in space [Bli82]. We use the color field $\varphi(\mathbf{x})$ described in Section 4.3.3 as a potential field. The guiding potential $\Theta_{s_i}^{\text{guide}}$ is defined as the squared distance from the position $\mathbf{s}_i$ of a surfel s_i to its projected position $\psi_I(\mathbf{s}_i)$ on an isovalue I of the potential field, i.e.,

$$\Theta_{s_i}^{\text{guide}} = \frac{1}{2}(\psi_I(\mathbf{s}_i) - \mathbf{s}_i)^2. \tag{6.22}$$

The projection $\psi_I(\mathbf{s}_i)$ is found using the Newton-Raphson method (see Section 2.4.2). Note that for efficiency reasons, we do not enforce this projection to be orthogonal. However, as the actual point-sampled surface is generally close to the isosurface, the difference is not significant. The normal $\mathbf{n}_{s_i}^{\text{guide}}$ at $\psi_I(\mathbf{s}_i)$ is equal to the gradient of the color field , i.e.,

$$\mathbf{n}_{s_i}^{\text{guide}} = \nabla\varphi(\psi_I(\mathbf{s}_i)). \tag{6.23}$$

Attracting Potential. Generally, we want the surface to coat the particles as tight as possible. Thus, a potential is defined such that the surfels are attracted to the particles. The attracting potential $\Theta_{s_i}^{\text{attract}}$ is computed as the sum of weighted squared distances from a surfel s_i to its neighboring particles $p_j \in \mathcal{P}_i$, i.e.,

$$\Theta_{s_i}^{\text{attract}} = \frac{1}{2}\sum_{p_j \in \mathcal{P}_i} (\mathbf{p}_j - \mathbf{s}_i)^2 \omega^{\text{poly}}(\mathbf{s}_i - \mathbf{p}_j, h_{\mathcal{P}}). \tag{6.24}$$

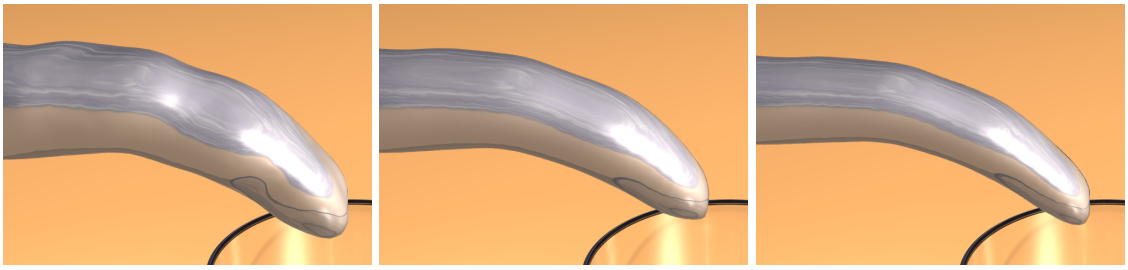


Figure 6.6: Illustration of the impact of the guiding, smoothing and attracting force. Left: guiding force only ($k^{\text{guide}} = 0.2$). Middle: guiding and smoothing force ($k^{\text{guide}} = 0.2$, $k^{\text{smooth}} = 0.6$). Right: guiding, smoothing and attracting force ($k^{\text{guide}} = 0.2$, $k^{\text{smooth}} = 0.6$, $k^{\text{attract}} = 0.1$).

Smoothing Potential. Minimization of the implicit and attracting potential yields the well-known blob artifacts due to the discretization of the volume with particles (see Figure 6.6). Hence, a potential Θ^{smooth} is defined that yields a smooth surface. The implicit surface described in Section 2.5.1 is a smooth surface Ψ_{smooth} that approximates the surfels. To ensure that only surfels of the same surface sheet are considered, the neighborhood for computing Ψ_{smooth} is restricted to neighbor surfels whose normals have an angle to the normal of s_i smaller than a threshold (we choose $\pi/4$). The smoothing potential $\Theta^{\text{smooth}}_{s_i}$ is then computed as the squared distance from s_i to Ψ_{smooth}

$$\Theta^{\text{smooth}}_{s_i} = \frac{1}{2}\left(\psi^{\text{orth}}(\mathbf{s}_i) - \mathbf{s}_i\right)^2, \tag{6.25}$$

where $\psi^{\text{orth}}(\mathbf{s}_i)$ is the orthogonal projection operator described in Section 2.5.1. The normal $\mathbf{n}^{\text{smooth}}_{s_i}$ on Ψ_{smooth} at $\psi^{\text{orth}}(\mathbf{s}_i)$ is

$$\mathbf{n}^{\text{smooth}}_{s_i} = \left(\mathbf{s}_i - \psi^{\text{orth}}(\mathbf{s}_i)\right) \frac{\left(\mathbf{s}_i - \psi^{\text{orth}}(\mathbf{s}_i)\right) \cdot \mathbf{n}_{s_i}}{\left\|\mathbf{s}_i - \psi^{\text{orth}}(\mathbf{s}_i)\right\|^2}. \tag{6.26}$$

Repulsion Potential. To achieve a locally uniform distribution, we define a repulsion potential Θ^{repel} similar to Pauly et al. [PGK02]. It is minimal when the neighboring particles $s_j \in \mathcal{S}^{\text{sub}}_i$ of a surfel s_i have a distance $h_{\mathcal{S}}$ to s_i:

$$\Theta^{\text{repel}}_{s_i} = \frac{1}{2} \sum_{s_j \in \mathcal{S}^{\text{sub}}_i} \left(h_{\mathcal{S}} - \left\|\mathbf{s}_i - \mathbf{s}_j\right\|\right)^2. \tag{6.27}$$

Minimizing Forces. The potential energy of the surfels s_i is minimized by apply-

ing forces that are derived from the energy fields

$$\mathbf{f}_{s_i}^{\text{guide}} = -\nabla_{\mathbf{s}_i}\Theta_{s_i}^{\text{guide}} = \psi_I(\mathbf{s}_i) - \mathbf{s}_i, \tag{6.28}$$

$$\mathbf{f}_{s_i}^{\text{attract}} = -\nabla_{\mathbf{s}_i}\Theta_{s_i}^{\text{attract}} = \sum_{p_j \in \mathcal{P}_i} \left(\omega_{s_i,p_j}\mathbf{r}_{\mathbf{p}_j,\mathbf{s}_i} - \frac{1}{2}\nabla\omega_{s_i,p_j}\mathbf{r}_{\mathbf{p}_j,\mathbf{s}_i}^2 \right), \tag{6.29}$$

$$\mathbf{f}_{s_i}^{\text{smooth}} = -\nabla_{\mathbf{s}_i}\Theta_{s_i}^{\text{smooth}} = \psi^{\text{orth}}(\mathbf{s}_i) - \mathbf{s}_i, \tag{6.30}$$

$$\mathbf{f}_{s_i}^{\text{repel}} = -\nabla_{\mathbf{s}_i}\Theta_{s_i}^{\text{repel}} = \sum_{s_j \in \mathcal{S}_i^{\text{sub}}} \frac{\left(h_{\mathcal{S}} - \left\|\mathbf{r}_{\mathbf{s}_i,\mathbf{s}_j}\right\|\right)}{\left\|\mathbf{r}_{\mathbf{s}_i,\mathbf{s}_j}\right\|}\mathbf{r}_{\mathbf{s}_i,\mathbf{s}_j}, \tag{6.31}$$

where $\mathbf{r}_{\mathbf{x}_i,\mathbf{x}_j} = \mathbf{x}_i - \mathbf{x}_j$ and $\omega_{s_i,p_j} = \omega^{\text{poly}}(\mathbf{s}_i - \mathbf{p}_j, h_{\mathcal{P}})$.

Similar to balloons [Coh91] and level sets [OS88], the surface is deformed only in normal direction. We thus restrict the guiding, smoothing and attracting potential to act only in direction of a surfel's normal, whereas the repulsion force only acts in tangential direction to achieve a locally uniform and hole free sampling. First, a new surfel normal $\mathbf{n}_{s_i}$ is computed as the average of the implicit gradient $\mathbf{n}_{s_i}^{\text{guide}}$ and the smoothed normal $\mathbf{n}_{s_i}^{\text{smooth}}$,

$$\mathbf{n}_{s_i} \leftarrow \frac{k^{\text{guide}}\mathbf{n}_{s_i}^{\text{guide}} + k^{\text{smooth}}\mathbf{n}_{s_i}^{\text{smooth}}}{\left\| k^{\text{guide}}\mathbf{n}_{s_i}^{\text{guide}} + k^{\text{smooth}}\mathbf{n}_{s_i}^{\text{smooth}} \right\|} \tag{6.32}$$

. The tangential force $\mathbf{f}_{s_i}^{\text{tan}}$ is then defined as the projection of the repulsion force $\mathbf{f}_{s_i}^{\text{repel}}$ onto the tangent plane defined by the new surfel normal $\mathbf{n}_{s_i}$:

$$\mathbf{f}_{s_i}^{\text{tan}} = \mathbf{f}_{s_i}^{\text{repel}} - (\mathbf{n}_{s_i} \cdot \mathbf{f}_{s_i}^{\text{repel}})\mathbf{n}_{s_i}. \tag{6.33}$$

The force in normal direction $\mathbf{f}_{s_i}^{\text{normal}}$ is computed as the projection of the implicit, attracting, smoothing and optional external forces

$$\mathbf{f}_{s_i}^{\text{normal}} = (\mathbf{n}_{s_i} \cdot \mathbf{f}_{s_i}^{\text{sum}})\mathbf{n}_{s_i}, \tag{6.34}$$

where

$$\mathbf{f}_{s_i}^{\text{sum}} = k^{\text{guide}}\mathbf{f}_{s_i}^{\text{guide}} + k^{\text{attract}}\mathbf{f}_{s_i}^{\text{attract}} + k^{\text{smooth}}\mathbf{f}_{s_i}^{\text{smooth}} + k^{\text{ext}}\mathbf{f}_{s_i}^{\text{ext}}. \tag{6.35}$$

The weights are user defined parameters that allow to trade smoothness for closeness of the surface to the particles, see Figure 6.6. The external force $\mathbf{f}_{s_i}^{\text{ext}}$ comes from an external force field and can be used, for instance, to attract the surface to a model surface (see Figure 6.12).

Integration. Finally, we get the new surfel position using explicit Euler integration as

$$\mathbf{s}_i \leftarrow \mathbf{s}_i + \alpha(\mathbf{f}_{s_i}^{\text{normal}} + \mathbf{f}_{s_i}^{\text{tan}}), \tag{6.36}$$

where $0 < \alpha \leq 1$ is the integration time step. Note that applying the forces along the new surfel normal vector $\mathbf{n}_{s_i}$ can be seen as a semi-explicit Euler integration, yielding a very stable integration if all weights are smaller than one, as $\mathbf{n}_{s_i}$ is

smooth (assuming that the isovalue I is chosen such that the isosurface is smooth). To avoid oscillations, we damp the system by multiplying α at each iteration with a damping constant. The integration is repeated until the maximal displaced distance of all surfels of a surface component is below a threshold.

Constraints

Constraints restrict the position and movement of the surface. We propose to use two constraints which are applied in the following order:

Guiding Constraint. Optionally, a surfel can be restricted to be within a minimal isolevel (see the guiding force description in the previous section), which is useful for splitting of solid objects as will be shown in Section 6.6.5. If the color field value $\varphi(\mathbf{s}_i)$ of a surfel s_i is smaller than a user defined minimal isovalue $I^{\min}$, then s_i is projected onto $I^{\min}$. This can also be seen as a maximum allowed distance to the particles.

External Constraints. External constraints are used to restrict the surface to a certain area, for example, when doing collision detection with an obstacle such as a glass. For collision detection with fixed boundaries only the particles are considered but not the surface. It can thus happen that surfels still penetrate the boundary although the particles do not. These penetrating surfels are projected onto the boundary. The projection is performed (if necessary) after each surface deformation iteration. Note that the surface remains smooth at the border of the projected surfels due to the smoothing potential.

6.6.3 Resampling

Resampling is important to adapt the number of surfels when the surface is stretched or compressed. Each time before a surface force is computed, we test if the number of neighbors $|\mathcal{S}_i|$ of a surfel s_i is smaller or larger than a minimum or maximum threshold, respectively. In the former case the number of missing neighbors are randomly distributed around s_i, where the new surfels inherit the neighbors of s_i. A neighbor update is then performed for the new surfels and the neighbors of s_i as described in Section 6.6.1. If the number of neighbors is too large, the surfel is deleted and removed from all its neighbors. We make the neighbor thresholds dependent on the radius of a surfel. Assume that all surfels have the same radius r_s and that the surfels are distributed uniformly, i.e., they lie on a hexagonal grid. Then a distance between two neighbors on this grid equal or smaller than r_s guarantees a hole free surface. We choose $6h_{\mathcal{S}}/r_s$ as the minimum threshold and $9h_{\mathcal{S}}/r_s$ as the maximum threshold.

This resampling scheme is also used to create an initial surface enclosing the particles, e.g., when using a source. Initially, a surfel is created at each particle position. These surfels are projected onto the implicit surface by setting the surface constants $k^{\text{guide}} = 1$ and $k^{\text{attract}} = k^{\text{smooth}} = 0$. Performing a few steps of surface

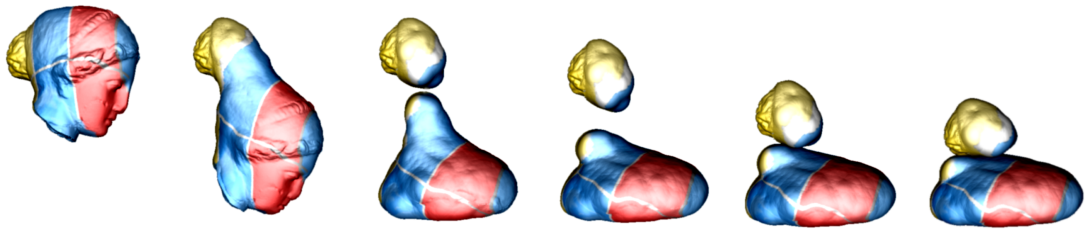


Figure 6.7: A melting model of Igea that splits. The split parts are detected and contact handling prevents them from merging again. So-called *zombie* surfels are used for interpolating the object texture.

deformation with resampling yields one or more closed surface components. Finally, the surface constants are reset to the user values, and applying the surface deformation scheme once again yields the final surface.

6.6.4 Zombies

An advantage of having a point-sampled surface is that every surfel carries its own information about both geometry and texture. During resampling, however, this information is distorted due to the movements and generation/deletion of surfels. Applying the surface update described in Section 6.6.2 yields the correct geometry information, however, the texture is distorted after a resampling step. To retain the texture, the initial surfels are copied and used for interpolating the color value. Because these copies are not part of the current geometry description, they are called *zombies* (see also [PKKG03]). Zombies are displaced as described in Section 6.6.2 and projected onto the new surface. The color of a surfel is then approximated from the neighboring zombies according to Equation (2.24). An example where zombies have been used is shown in Figure 6.7.

6.6.5 Topological Changes

By recomputing the neighborhood and using the forces and constraints described above, the surface implicitly handles topological changes such as disjoint components and merging (see Figure 6.8). The implicit constraint ensures that a (locally) stretched surface is always split even if the implicit force weight is chosen to be small. Furthermore, two intersecting surface components are merged automatically by recomputing the surfel neighborhood and the applied forces. A well known problem of handling topological changes implicitly is that two surface components will blend rather than collide, i.e., they are merged before they intersect which results in considerable artifacts. We suggest a method to detect disjoint surface components similar to the blending graph described by Desbrun

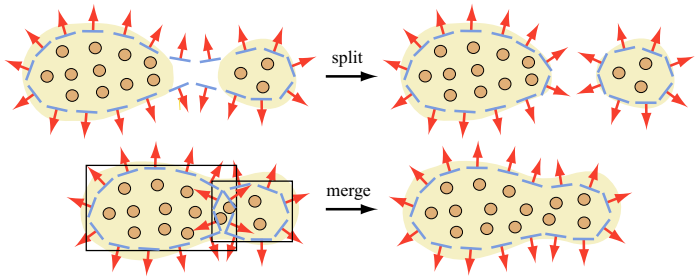


Figure 6.8: Top row: when a surfel is outside the minimal isovalue it is projected onto it. Flood filling over all the surfels results in the construction of separated components. Bottom row: merging is performed by detecting and removing colliding surfels from different components.

and Cani [DC95]. These components are then handled as separated objects. We show how intersecting separated objects are merged.

Splitting

To detect two disjoint components of a surface after having deformed the surface, a flood-fill is performed over all surfels in the restricted surfel neighborhoods $\mathcal{S}_i^{\text{rest}}$. A neighbor $s_j \in \mathcal{S}_i^{\text{sub}}$ is added to $\mathcal{S}_i^{\text{rest}}$ if the angle between its normal and the normal of s_i is smaller than a threshold (we choose $\pi/4$). Starting with an arbitrary surfel s_i, we add s_i and the surfels in $\mathcal{S}_i^{\text{rest}}$ to a set $\mathcal{S}^{\text{sep}}$. The neighbors of the restricted neighborhood of the surfels in $\mathcal{S}_i^{\text{rest}}$ are then added recursively to $\mathcal{S}^{\text{sep}}$, until no neighbors are left. This procedure is repeated (with new sets) as long as there are surfels that do not belong to a set yet. By tagging surfels that belong to a set already, the detection can be done in linear time to the number of surfels, assuming a constant maximum number of neighbors.

After separating the surfels, the particles are assigned to the appropriate set by performing an inside/outside test, see Equation (5.35). A particle is added to a set if it is inside the surface represented by the surfels. Each set then builds a separated surface component. The surfel neighbors $\mathcal{S}_i$ and the particle neighbors $\mathcal{P}_i$ of a surfel s_i are always computed from the surfel and particle set of its separated component.

Merging

When two disjoint components intersect they either need to be merged or contact handling has to separate them as described in Section 5.8. For merging we require that not only the surfaces intersect, but also that at least one particle is inside the other surface. This guarantees that the surfaces are merged smoothly.

We first compute a bounding box for each object surface part (see Figure 6.8).

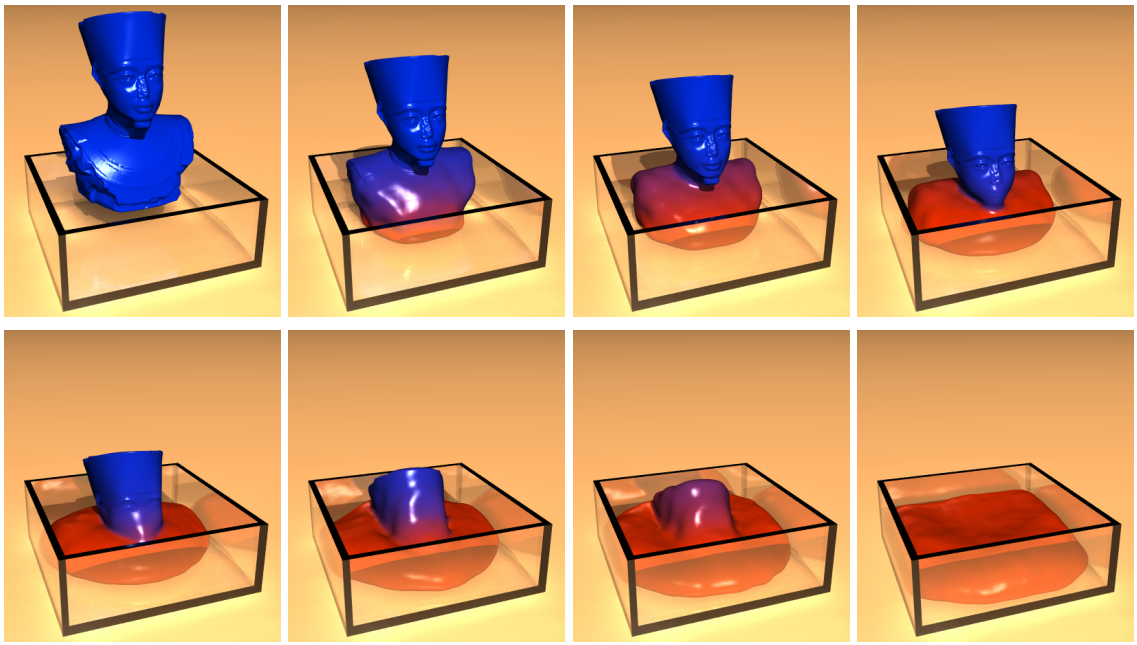


Figure 6.9: An elastic solid is dropped onto a heated box and slowly melts to a viscous fluid. 3.9k particles, 56k surfels, 25s/frame

From two intersecting bounding boxes, the colliding particles can be efficiently computed (see Section 5.8.2). If a set of colliding particles is found, the penetrating surfels are computed and deleted before the two objects are merged. The surfel neighborhoods are then recomputed and the separated objects merge smoothly through the acting surface forces. Note that this way we avoid the unnatural blending typically arising when using the implicit function for merging, i.e., blending of two separated surface parts before they intersect is avoided.

6.6.6 Blending Between Solids and Fluids

Whereas solids often have a very detailed surface, fluid surfaces are usually rather smooth. The particles account for this by surface tension (cohesion, see Section 4.4.2). However, to smooth the surface this is not sufficient. Assume we start with a highly detailed solid that melts. In this case we expect the detail to disappear. On the other hand if we freeze a fluid, the existing detail should be preserved.

If we only apply the surfel displacement according to the particles, all the detail is preserved, but if we additionally update the surface using the potential fields, the detail vanishes and the surface approaches the implicit surface defined by the particles. To get a smooth transition between solids and fluids, we perform both approaches and blend between them.

Assume that after particle animation, a surfel s_i is displaced to the position

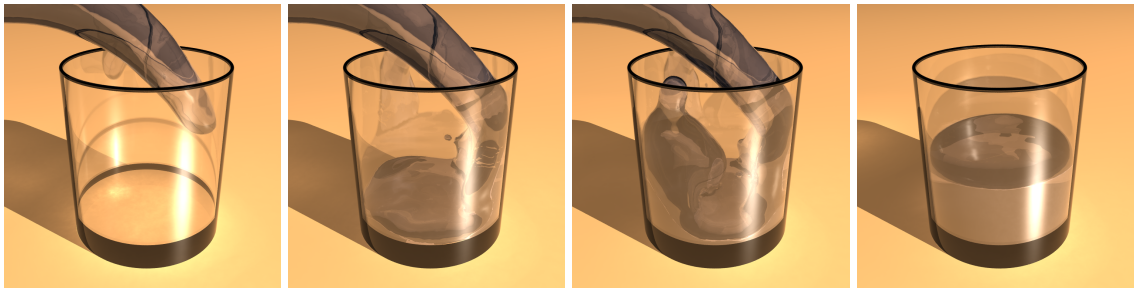


Figure 6.10: Pouring a pure fluid into a glass. 3k particles, 3.4k surfels, 1.4s/frame.

$\mathbf{s}_i$. Let $\mathbf{s}'_i$ be the positions after applying the surface deformation forces and constraints. We get the blended position $\mathbf{s}_i^{\text{blend}}$ and normal $\mathbf{n}_{s_i}^{\text{blend}}$ by simple interpolation

$$\mathbf{s}_i^{\text{blend}} = (1-\beta_i)\mathbf{x}_{s_i} + \beta_i \mathbf{s}'_i, \tag{6.37}$$

$$\mathbf{n}_{s_i}^{\text{blend}} = \frac{(1-\beta_i)\mathbf{n}_{s_i} + \beta_i \mathbf{n}'_{s_i}}{\left\|(1-\beta_i)\mathbf{n}_{s_i} + \beta_i \mathbf{n}'_{s_i}\right\|}. \tag{6.38}$$

For melting and freezing, we use the temperature T_{s_i} as a blending factor (see Section 6.2.1). The temperature of a surfel is approximated from the neighboring particles

$$T_{s_i} = \frac{1}{\sum_{p_j \in \mathcal{P}_i} \omega^{\text{poly}}(\mathbf{s}_i - \mathbf{p}_j, h_{\mathcal{P}})} \sum_{p_j \in \mathcal{P}_i} \omega^{\text{poly}}(\mathbf{s}_i - \mathbf{p}_j, h_{\mathcal{P}}) T_{p_j}. \tag{6.39}$$

The normalized temperature is then used as a transition factor, i.e., $\beta_i = (T_{s_i} - T^{min})/(T^{max} - T^{min})$, where T^{max} and T^{min} are the temperature thresholds described in Section 6.2.1.

6.7 Results

We have tested our physics framework and the surface generation on a variety of examples that demonstrate melting, freezing and the different elasto-plastic and viscoelastic behaviors depending on the parameters.

In the first example we have a pure fluid (no elastic forces) to test our surface reconstruction. Our surface is able to handle all topological changes like splitting, merging, and self-intersections of the surface. Merging of disjoint surface components prevents from blending two components before they intersect (see Section 6.6.5), resulting in less artifacts compared to implicit merging.

Two examples of solidification are shown in Figures 6.1 and 6.11. In the former example, we show the fluid particles created by a source. When the particles get in contact with the ground, their temperature cools down and they solidify.

Figure 6.11: Freezing a quicksilver fluid that is poured into a glass. After removing the glass, the elastic solid bounces onto the ground and fractures. 2.4k particles, 3.4k surfels, 2s/frame.

The ground has a temperature of 0°C and the initial particle temperature is 20°C. Heat diffusion between particles is $k^{\text{heat}} = 0.5$ and between ground and particles $k_{\text{ext}}^{\text{heat}} = 1$. The particle temperature locally affects the stiffness, plasticity, viscosity and surface tension of the object as described in Section 6.5. Particles with a temperature of 0°C, shown in blue, are completely solid, whereas particles with 20°C, shown in red, are completely fluid ($E = 0$).

Figure 6.11 shows a quicksilver-like fluid ($\mu = 10\text{kNs/m}^2$) that is poured into a glass. While it is flowing the temperature is decreased of all particles within 10 iterations from $T^{\max}$ to $T^{\min}$. Therefore, the fluid freezes to an elastic solid with stiffness $E = 500\text{kN/m}^2$. At the same time we remove the glass. The solidified fluid elastically bounces onto the ground where it splits.

Three examples demonstrate the versatility of our framework for melting. In Figure 6.3, a solid deformable cube with 14^3 particles is dropped into a heated funnel where it melts. When it hits the cool ground it solidifies again. Two simulations with different plasticity are shown, where the plastic strain ε^{p} in the simulation shown on the left is twice as big as in the one shown on the right hand side. The images are taken at corresponding time steps. Naturally, the object with higher plasticity flows faster through the funnel, whereas the cube on the right is more rigid and therefore takes longer until it melts. The different behavior can also be seen once the melted object solidifies again on the ground. The one with higher plasticity distributes more, whereas the one on the right stays more compact.

In the example shown in Figure 6.9, we start with an elastic bust of the Nefertitis (57k surfels) which is dropped onto a heated box. When a particle collides with the box, its temperature increases and diffuses to the other particles ($k^{\text{heat}} = 0.04$,

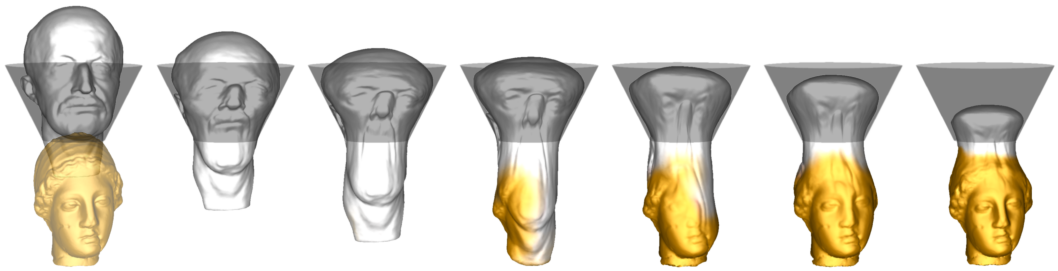


Figure 6.12: A melting model of Max Planck flows through a funnel into the Igea casting mold. The melting surface is attracted by the mold by an external surface force field.

$k_{\text{ext}}^{\text{heat}} = 1$). The model first bounces elastically and then slowly melts to a viscous fluid ($\mu = 10\text{kNs/m}^2$). Note that the surface detail is still preserved even though large parts of the model are already liquid. The phase state of the Nefertitis is color coded from blue for solid to red for fluid.

In Figure 6.4 we show an example with an initial surface with sharp edges. The left side of the cube is heated immediately, with low heat diffusion. As expected, the edges of the heated part smooth out during melting, whereas the edges on the right hand side are preserved. The cube is sampled with 15^3 particles and the initial surface with 3.9k surfels. Note that the rippling artifacts in the cube come from the coarse particle resolution and the regular particle sampling (see Figure 6.4 upper row).

Three viscoelastic fluids with different stiffness E are compared in Figure 6.13. 2.5k particles are sprayed on a glass wall with a rate of 3.5k particles/s and an initial velocity of 2.5m/s. The viscosity of the fluids is set to $\mu = 4\text{kNs/m}^2$. The fluid on the left has no restoring elastic forces, i.e., $E = 0$. The fluid splashes into many drops and droplets due to the collision with the wall. The large number of droplets cause a high increase in surface area, resulting in up to 45k surfels used during the simulation. Nevertheless, it demonstrates the stability of our surface extraction method. For the fluid in the middle we chose Young's modulus to be $E = 400\text{kN/m}^2$. The fluid does not splash anymore but stays very compact and sticky. Finally, for the simulation shown on the right we set $E = 900\text{kN/m}^2$. The restoring elastic forces are strong enough to make the fluid stiff. The fluid therefore folds there where it hits the boundary. After all fluid is out of the source, it elastically drops onto the ground.

Our last examples demonstrate the versatility of our surface animation approach. In the example shown in Figure 6.7 the shock of hair of the Igea model

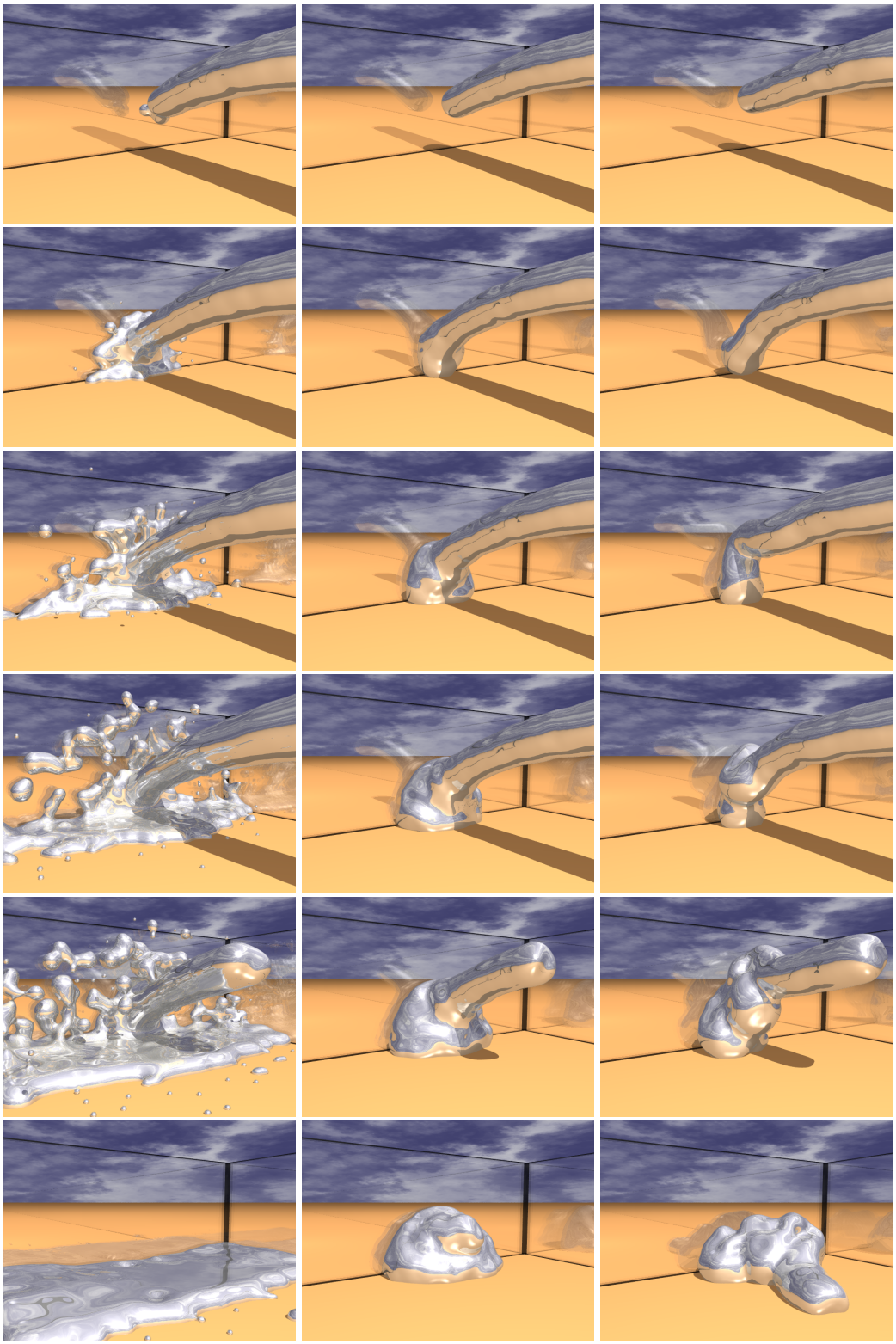


Figure 6.13: Comparison of viscoelastic fluids. Left: purely viscous fluid (no restoring elastic forces). Middle: fluid with median stiffness. Right: stiff elastic fluid.

Animation	Particles/Surfels	Physics	Surface	fps
Fluid	3k/3.4k	0.13 s	1.3 s	0.7
Freezing	2.4k/3.4k	0.4 s	1.2 s	0.5
Melting	3.9k/56k	3.1 s	20 s	0.03

Table 6.1: Average timings for the sequences shown in Figures 6.10, 6.11 and 6.9, running on a 3 GHz Pentium 4 with a GeForce FX GPU. Timings are shown for one physics animation step and one surface deformation iteration step, followed by the resulting frame rate.

is fixed initially. Due to gravitation, part of the model splits off and drops to the plane. The guiding constraint (Section 6.6.2) restricts the distance of the surfels to the particles, therefore the surface also splits naturally. The split surface parts are detected automatically (Section 6.6.5). Afterwards, the shock of hair is released. The two split surface parts are treated as separate objects. Thus, contact handling (Section 5.8) prevents the two parts from merging again. Note that the texture of the model is preserved due to the interpolation of zombies (Section 6.6.4). Initially, the Igea model has 134k surfels. This number increases to 340k surfels during the animation due to topological changes. At the end of the animation sequence, the number of surfels is 165k.

The second example shown in Figure 6.12 exploits an external surface force. A highly plastic model of Max Planck with 53k surfels and sampled by 600 particles flows through a funnel into the Igea model (134k surfels). The Igea model defines an external surface force field that attracts the surfels (Section 6.6.2). Thus, the melted Max Planck fills the fine surface detail of the Igea mold. At the end the model is solidified and exhibits realistic elastic behavior when it is dropped onto the ground. Due to the resampling (Section 6.6.3), the final solidified model consists of 115k surfels.

The physics animation runs in interactive time for up to 3900 particles, see Table 6.1. The performance of the surface generation algorithm depends on the surface resolution and the surfel neighborhood radius $h_{\mathcal{S}}$. For the fluid example shown in Figure 6.10 (average number of 3.4k surfels) our algorithm needs on average 1.3 seconds per frame. For the same example with smaller neighborhood (average number of 1.2k surfels) the animation runs with 1.5 frames per second. We use a low resolution surface for interactive animations and prototyping, and increase the resolution for making production animations. The pictures in this chapter were created with the open-source renderer POV-Ray (`http://www.povray.org`), which we modified for raytracing point-sampled objects.

6.8 Limitations & Future Work

Both your meshless physics framework and the point-based method for surface extraction showed to be versatile and enable the simulation of viscoelastic material as well as melting and solidifying of objects. Nevertheless, there are many possible improvements and open problems for future research.

Using a meshless volume representation, the rest shape can easily adapt to the deformed shape as discussed in Section 6.4.1, which highlights one of the advantages of using a representation with no explicit connectivity. However, a disadvantage of this update is that the original shape information is lost, yielding plastic deformations. Therefore, for stronger non-plastic deformations this approach is not suitable. Although it is possible to change at run time from a rigidly transformed rest shape to a rest shape that is deformed every time step and vice versa, so far we can only use one approach for the whole object. It is therefore difficult to model an object that is elastically deformed at one place and melts at another. An approach could be to compute the forces for both the rigid and the deformable rest shape, and compute a weighted average between them according to particle temperature (see Section 6.5). More advanced methods for computing a suitable rest shape need to be explored.

The heat transfer model and especially the change of physical parameters depending on the temperature is rather ad hoc and not according to a physical model. In the future, a more accurate model should be derived from the physics literature. However, once a suitable model has been identified, it will be easy to incorporate it into this framework.

Although the point-based surface model is very flexible, there are also several disadvantages of this representation. To represent small drops in a strongly splashing fluid a prohibitive high number of surfels with small radii are needed. Instead of using the same surfel sizes everywhere, the density of the surfels could be chosen depending on whether they lie on a flat or on a highly curved surface. A point-based representation is most suitable where the topology of the object changes frequently, otherwise using a mesh representation can be advantageous. An interesting direction for future research is therefore to use a combination of both. Also the representation of sharp creases is more difficult with a point-based representation. In Section 5.5 creases and corners are handled by the renderer using CSG between surfels. This is more difficult to apply in this dynamic surface model because surfels change their position after resampling. Surfels on an edge would therefore not be allowed to move, while new surfels that overlap the edge also have to be clipped. Depending on the melting state and the angle between two corresponding surfels, the edge would finally disappear.

Because the surface update is based on an implicit model (Section 6.6.2), the surface is always consistent and intersection free. However, the surface displacement approach might yield self-intersections. In this case, the blended surface might have self-intersections as well. Generally, this hybrid implicit-explicit

model might cause problems during blending if the topology of the implicit and explicit surface are not similar. A model that avoids blending all together would be desirable.

Implicit blending artifacts between two objects that come close are avoided by separating disjoint surface components and merging them explicitly when two components intersect (Section 6.6.5). However, these artifacts still occur when surface parts come close that have not been split. Similar to the surface consistency problem described above, detection of self-intersections would be necessary which is a very difficult problem.

A well-known problem in SPH based animations is volume preservation. Furthermore, also our surface generation approach is not volume preserving. This could be improved by adding an additional force that lets the volume shrink or grow, similar to the volume preservation constraint suggested by Desbrun and Cani [DC98].

So far we use a global termination criterion for the iterative integration of the surface forces for surface deformation (see Section 6.6.2). Because the forces are defined locally, efficiency could be improved significantly by computing and integrating the forces only for surfels s_i whose position changed and for the neighbors of s_i.

6.9 Summary

In this chapter a method for simulating viscoelastic materials has been derived which proved to be versatile and stable. It enables modeling a variety of materials that could not be simulated before, like elastic fluids, fluids that solidify, and solids that melt to splashing fluids. Melting and freezing are modeled by locally changing the material parameters according to the particles' temperature.

The presented hybrid implicit-explicit point-based surface model fits the needs to represent both highly detailed solids and smooth fluids with rapidly changing topology. Surfels showed to be suitable as an explicit representation because no mesh needs to be maintained.

Exciting research directions are the derivation of a more accurate physical model for melting and freezing, and exploring a new surface model. The surface representation needs to guarantee temporal and topological consistency, while it is also suitable for modeling sharp features and detailed geometry. Furthermore, both real-time and high quality surface animation should be possible with the same surface model, for instance, in a progressive manner using a multiresolution approach.

Chapter 7

Implementation

In this chapter a discussion and implementation details are provided of the search data structures we use to speedup neighbor queries (Section 7.1), and of the time integration schemes applied to solve the equations of motion (Section 7.2).

7.1 Search Data Structures

The Smoothed Particle Hydrodynamics method (Section 2.3) as well as operations defined on a point-based surface representation (Section 2.5) are meshless methods. Both rely on local operators based on the spatial proximity between points (particles or surfels) instead of the geodesic proximity given by the mesh (e.g. tetrahedra or triangles). Thus, performing *range queries* (also called *distance queries*), i.e., finding all points within a distance h to a point $\mathbf{x} \in \mathbb{R}$, is very important and in fact most often the performance bottleneck of a simulation (see Section 4.7.1 for a discussion and timings). Searching for neighbors within a certain distance h sequentially needs time $O(n)$, where n is the number of data points. Several data structures have been proposed to reduce this time. Examples are (with increasing complexity) *uniform grids*, *hash grids*, *octrees*, *kd-trees* and *bsp-trees*, and combination of different data structures (see Samet [Sam89,Sam05] for a thorough discussion). These data structures differ in the computational complexity for building or updating it and performing range queries, as well as in memory consumption. Non-hierarchical data structures such as grids can be build and updated very efficiently, whereas hierarchical data structures such as octrees, *k*d-trees and bd-trees are usually more expensive to build or change, but often allow faster queries than grids. However, the performance depends largely on the number and distribution of the data points, as well as the queries.

We will next discuss the two search data structures that are used in this dissertation, namely hash grids (Section 7.1.1) and *k*d-trees (Section 7.1.2).We then compare them and try to give a statement in what situation which search data structure is preferable (Section 7.1.3).

7.1.1 Hash Grid

A hash grid can be seen as a uniform grid, where each grid cell (an axis-aligned box) is mapped onto a 1D hash table index. Given a 3D grid cell with size l, a point $\mathbf{x} = [x,y,z]^T$ is mapped onto an index j as follows

$$j = f(\lfloor x/l \rfloor, \lfloor y/l \rfloor, \lfloor z/l \rfloor), \tag{7.1}$$

where f is a hash function and $\lfloor\rfloor$ rounds the coordinates down to integer values. Data points are stored in the hash grid by mapping them onto j and storing them in a hash table in the bucket at index j. Note that building a hash table is very efficient: data points can be inserted in constant time $O(1)$, resulting in a building complexity of $O(n)$ for n data points.

The mapping has the advantage compared to a uniform grid [Hec97] that the hash grid is defined over the whole space. The mapping is non-injective, meaning that different grid cells can be mapped to the same index. Thus, the bigger the table size (the more buckets) the less likely it is that two or more cells map onto the same index (this is called a *collision*). Because collisions decrease the computational performance, a trade-off between the number of collisions and memory management (especially memory caching) due to the size of the table has to be made. Since only cells containing at least one data point are mapped, the table size is independent of the spatial size of the domain spanned by the data points but depends only on the number of data points and the size l of a grid cell. Best performances are achieved if the hash table size is a prime number [CLR90] and significantly larger than the number of data points [THM+03].

The number of collisions depends besides the number of buckets also on the size of a cell and the distribution of the data points. With bigger cell size, i.e., less cells, the chance of a collision decreases, but at the same time a cell contains more data points that are mapped to the same bucket and need to be processed.

A *perfect hash table* that guarantees that there are no collisions has been presented recently by Lefebvre and Hoppe [LH06]. However, due to the long computational time needed for building the hash table it is not suitable for dynamic data.

In our implementation the hash function of Teschner et al. [THM+03] is used:

$$f(i,j,k) = (i \cdot p_1 \ \textbf{xor}\ j \cdot p_2 \ \textbf{xor}\ k \cdot p_3) \ \textbf{mod}\ m, \tag{7.2}$$

where $p_1 = 73856093$, $p_2 = 19349663$ and $p_3 = 83492791$ are large prime numbers and m is the hash table size. Given a maximum query distance $h_{\max}$, we choose the cell size l to be at least $h_{\max}$. This has the advantage that for any query we need to search only the cell that contains the query point and its 27 neighboring cells. This can be implemented very efficiently. First, the query point is mapped onto a bucket. For the data points in this bucket and in the neighboring cells that intersect the query ball, the squared distance to the query point is tested to filter out data points that are further away than the query distance h. Note that

this has to be done even for cells that are completely inside the query ball because of possible mapping collisions.

Assuming that there are no collisions, the expected time complexity for such a query is $\Theta(q)$, where q is the average number of data points in a cell. If the distribution of the data points is about uniform, then the time complexity is even constant.

7.1.2 *k*d-Tree

A *k*d-tree (short for *k*-dimensional tree) is a special case of a binary space partitioning tree (BSP-tree), originally developed for indexing multi-dimensional data point sets in higher dimensions [Ben75, FBF77, AM93]. The root cell (an axis-aligned box) contains the whole space $\mathbb{R}^3$. In every level of the hierarchy, an axis-aligned plane splits a cell into two sub-cells, where the split axes are cycled when moving down the tree. The leaf nodes are called buckets and contain at least one data point. Thus, the union of all buckets represents the whole space.

Because the position of the splitting plane can be chosen arbitrarily within a cell, the *k*d-tree can partition the space better than a (hash) grid or an octree. This is especially important if the distribution of the data points is very uneven. If the split plane is chosen such that it goes through the median of the data points of a node, i.e., cuts them into equal halves, the tree is balanced and thus has height $O(\log n)$ with n the total number of data points. The time complexity for building the *k*d-tree is $O(n \log n)$ and takes $O(kn)$ space with k the number of dimensions. A point can be inserted into or deleted from a balanced *k*d-tree in time $O(\log n)$, however, efficient rebalancing the *k*d-tree afterwards is a notoriously difficult task.

For range queries of a point $\mathbf{x} \in \mathbb{R}^3$, the tree is traversed from the root until the bucket containing $\mathbf{x}$ is found, where cells can be discarded if they do not intersect the query ball. The distance of a cell to the query point can be efficiently computed using an incremental update while traversing down the tree as shown by Arya and Mount [AM93]. From a leaf node, the data points within the query ball are determined by backtracking towards the root. Finding the leaf node can be done in $O(\log n)$, whereas the time for backtracking is on average roughly proportional to the number j of found data points, yielding an average time complexity of $\Theta(j + \log n)$ (see Chancy et al. [CDZC01] for a careful analysis). The worst time query complexity is $O(n^{1-1/k} + j)$, see e.g. Lee and Wong [LW77].

We use a *k*d-tree similar to [Mou05], where we build the *k*d-tree with a top-down approach using the *Sliding-Midpoint* rule [AF00] and store the data points only in the leaf nodes. Details and an analysis of your implementation can be found in [Kei03].

7.1.3 Discussion

As already discussed above, the choice of the best search data structure depends on the query and the size, distribution and dynamics of the data. A hash grid is build in linear time, and therefore faster than building a *k*d-tree which takes time $O(n \log n)$. Furthermore, updating (inserting, deleting or moving a data point) of hash grids is usually fast, whereas rebuilding a *k*d-tree from scratch is often faster than updating it [Rob81]. Thus, it is often proposed to use a *k*d-tree for static data and a hash grid if the data changes dynamically. However, a *k*d-tree has many advantages even in the dynamic case. Generally, a query is faster using a *k*d-tree. In our experiments a *k*d-tree query in 3D clearly outperforms the hash grid with large number of data points. We assume that this is because the hash grid cannot profit as much from cache memory anymore as with fewer data points. Furthermore, with larger data sets and many queries, the time for querying is much higher than building the *k*d-tree, thus rendering the building time insignificant as shown in our experiments described in Section 4.7. Another advantage of *k*d-trees is that we do not have to know the query distance when building the data structure. Even more important, arbitrary query distances can be used as, for instance, in our multiresolution framework (Chapter 4). Remember that for the hash grid the cell size is set equal to the largest query distance, thus decreasing the performance for queries with a much smaller distance. Finally, a *k*d-tree can be used for j nearest neighbor queries with the same time complexity as for range queries. Especially, a *k*d-tree is very efficient in finding the closest neighbor to a point.

Thus, we can state that a hash grid should be used for interactive applications, where the number of data points and queries is small (a few thousands) and therefore the costs for building the data structure matters. Furthermore, hash grids have the advantage that they can be implemented exploiting hardware (see [Sig06] for a description of a two-layered hash grid implemented on the GPU). However, the size of the hash grid is restricted to fit into memory. For larger data sets and many queries, we recommend to use a *k*d-tree because the building time is insignificant compared to the time needed for queries. So far, we do not know any implementation of *k*d-trees exploiting hardware.

7.2 Time Integration

To compute the position, velocity and other properties of the particles and surface at a certain point in time, we need to integrate the equation of motion in time. As already discussed in Section 2.1.1, this equation can be rewritten as a coupled set of two first order equations

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{7.3}$$

$$\dot{\mathbf{v}} = f(\mathbf{v}, \mathbf{p}, t), \tag{7.4}$$

where $\mathbf{p}$ is the position of a particle, $\mathbf{v}$ its velocity, and $f()$ a general function given by the physical model.

Various methods exists for numerical time integration, see [HES03] for a nice overview in the context of deformable modeling in computer graphics. Generally, the techniques are classified either as *explicit* or *implicit* integration depending on whether the unknown quantities are computed considering the current state only, or whether they are implicitly given as the solution of a system of equations involving both the current state and the unknown state of the system, respectively. While explicit methods are easy to implement and fast to compute, they are only conditionally stable depending on the chosen time step Δt. Furthermore, explicit integration is only convergent if the time step fulfills the *Courant-Friedrichs-Lewy (CFL) condition* [CFL28]. On the other hand, implicit time integration is stable for arbitrary time steps, but it requires solving an algebraic system that is often non-linear. Furthermore, time steps that are too big yield overdamping of the system.

7.2.1 Leapfrog Integration

We found the leapfrog scheme [FLS63] to be the most attractive explicit integration scheme due to its simplicity and second-order accuracy. In this scheme the velocities are evaluated at the midpoint of the time intervals, thus they are staggered with respect to the positions

$$\mathbf{v}_i(t+\frac{1}{2}\Delta t) \leftarrow \mathbf{v}_i(t-\frac{1}{2}\Delta t)+\Delta t\frac{\mathbf{f}_i(t)}{m_i}, \tag{7.5}$$

$$\mathbf{p}_i(t+\Delta t) \leftarrow \mathbf{p}_i(t)+\Delta t\mathbf{v}_i(t+\frac{1}{2}\Delta t), \tag{7.6}$$

where $\mathbf{v}_i(t)$ is the velocity of a particle p_i at time t, $\mathbf{p}_i(t)$ its position, $\mathbf{f}_i(t)$ the total force acting on p_i, and m_i its mass. This integration scheme is as simple as the first-order accurate explicit Euler integration, but is second-order accurate. Furthermore, it has the very nice property that it is symmetric and thus *time reversible* [McM06], i.e., we can follow the trajectory of a particle back in time as follows

$$\mathbf{v}_i(t-\frac{1}{2}\Delta t) \leftarrow \mathbf{v}_i(t+\frac{1}{2}\Delta t)-\Delta t\frac{\mathbf{f}_i(t)}{m_i}, \tag{7.7}$$

$$\mathbf{p}_i(t-\Delta t) \leftarrow \mathbf{p}_i(t)-\Delta t\mathbf{v}_i(t-\frac{1}{2}\Delta t). \tag{7.8}$$

This backtracking is precise, i.e., if a particle starts at a point and is integrated n time steps forward and then n time steps backward in time, the particle arrives exactly at the starting point. Note that this is not the case for other integration schemes such as explicit Euler, midpoint or fourth-order Runge-Kutta integration. Time reversibility is important in physics because only then conservation of energy and momentum is guaranteed. But also in computer graphics this is very

useful as it allows to start at an arbitrary point in time and then compute the simulation forward and backward. For instance, a designer can first manually set up the desired state of a system such as a breaking wave, or use a time reversed fluid animation that flows up a stair to control another fluid simulation as shown in [TKPR06]. A drawback of the leapfrog technique is that the velocity at time t is not given explicitly. Where needed we compute it as the average of the previous and following velocity

$$\mathbf{v}(t) = \frac{1}{2}\left(\mathbf{v}_i(t - \frac{1}{2}\Delta t) + \mathbf{v}_i(t + \frac{1}{2}\Delta t)\right). \tag{7.9}$$

Furthermore, an additional computation step is needed to compute the initial velocity $\mathbf{v}_i(\frac{1}{2}\Delta t)$, for instance, using explicit Euler integration.

7.2.2 Discussion

For simulating deformable bodies we implemented both an explicit (*leapfrog*, see above) and implicit (*implicit Euler*, see [MKN+04]) time stepping scheme. Although larger time steps can be used for implicit time integration, solving the linearized equation system showed to be more expensive then performing several iterations of explicit integration with a smaller time step. Furthermore, smaller time steps are advantageous for contact handling due to smaller penetrations. Because of the big time steps we use, implicit integration tends to smooth out fine scale wobbling of soft material, which, however, is advantageous when simulating stiff objects. In our system, where we combine fluids and elastic solids, the explicit leapfrog scheme gives a good trade-off between stability, efficiency and damping. However, choosing the right integration scheme and time steps is extremely important for the stability and accuracy of a simulation as demonstrated by Hauth et al. [HES03]. They present a simple variant of Newton's method coined *inexact Newton's method* [Rhe98] for efficiently solving non-linear equation systems, and use the number of Newton iterations to adapt the time step. Recently, Kharevych et al. [KYT+06] proposed geometric, variational integrators of arbitrary accuracy-order that are efficient and preserve momentum and energy. In future work, we want to experiment with different explicit and implicit time integration schemes and compare their accuracy to a reference solution. Furthermore, adaptive time stepping could yield significant speedups. Additionally, time steps can also be adapted to the resolution to even further improve the performance of multiresolution simulations, as shown by Desbrun and Cani [DC99].

Chapter 8

Conclusion

This chapter concludes the dissertation with a summary of the main contributions and achieved results, a discussion of the advantages and drawbacks of the presented approaches, and possible directions for future research.

8.1 Summary

The objective of this dissertation is to explore meshless Lagrangian methods for physics-based animation of fluids and solids and simulating their interactions, and the use of point-based surface representations in this context.

Smoothed Particle Hydrodynamics has been used as a numerical approximation method for solving the Newtonian equations of motion for fluids and deformable solids. New contributions to the computer graphics community in this context are:

- a particle-based multiphase approach for two-way coupling of immiscible fluids of arbitrary density. Fluids are separated by **modeling cohesion as a pressure jump** at the interface. We exploit this approach to simulate coupling between water and air, including the animation of bubbles as trapped air.
- a **derivation of elastic forces** from a model based on continuum mechanics. A first order approximation of the displacement field is computed using the moving least-squares approach. This results in stable animations of deforming elasto-plastic objects that are invariant under rigid body motion and run at interactive rates for models with moderate complexity.
- Application of the transparency method to **model discontinuities in the smooth shape functions**. This enables the simulation of discontinuities in the phyiscal domain introduced by fracture surfaces. Fracturing of both brittle and ductile material has been achieved, as well as controlled fracture.
- a **multiresolution** method based on the concept of virtual particles to achieve consistent coupling between different resolution levels and to dy-

namically adapt the resolution to the simulation characteristics. This approach proved to be stable and has very little computation and memory overhead. For our fluid simulations, a performance gain up to a factor of six has been achieved. Similarly, the discretization of the physical domain is adapted to handle fracturing and large deformations of deformable objects.

- **enhancing fluids with elastic forces** to simulate viscoelastic materials. By introducing an adaptive rest shape that is used to compute restoring elastic forces, materials in the range of stiff elastic to elasto-plastic and viscoelastic can be simulated. Furthermore, melting and freezing effects have been achieved by locally adapting the material properties.
- **solid-fluid interaction based on a unified particle metaphor.** Solids are treated as (rigid) fluids, i.e., the same forces as for multiphase fluids are applied to solid particles. Additionally, collision handling between rigid bodies is based on forces derived from molecular dynamics, which act between particles of different objects. Thus, interaction handling between solids and fluids is purely particle-based without considering the surface, yielding stable and fast simulations.

A surface sampled with oriented point samples (surfels) has been used to animate and deform the boundary of deforming objects and fluids. New contributions in the context of surface modeling for animations are:

- a **free-form deformation** approach for advecting an embedded surface with the particles. A first order approximation of the displacement field from the neighboring particles is computed at the surfel positions to determine their displacement from the rest shape. This approach is suitable for fast deformations of geometrically detailed surfaces that do not undergo topological changes.
- **surfel splitting** for efficiently adapting the surface sampling to avoid surface distortions for large deformations.
- a **contact handling** scheme for deforming objects that decouples collision detection and deformation. A non-penetrating contact surface is computed from the detected colliding surfels and sampled with contact nodes. Penalty forces are then computed for each contact node and distributed to the particles. This decoupling of surface collision detection and volumetric deformations results in efficient and stable contact handling.
- a **crack model** for generating fracture surfaces and handling of topological events. Propagating cracks are dynamically sampled by adding surfels. Termination, splitting and merging of cracks are detected and handled. With our simple scheme, complex crack patterns and detailed fracture surfaces can be achieved.
- the **derivation of potential fields** to dynamically adapt the surface to the physics characteristics. Forces acting on the surfels are computed from the

potential fields that guide the surface deformation and enable the implicit handling of topological changes. Smoothness and distance of the surface to the particles can be controlled by the user. This approach yields stable surface animations for both deformable objects with geometrically detailed surfaces, and object animations with smooth surfaces and rapid topological changes such as splashing fluids. Furthermore, it enables the (local) transition from detailed solid surfaces to smooth fluid surfaces while melting.

- surface **texture preservation** for extreme deformations and melting by advecting the texture information. Copies of the original surfels, so-called zombies, are animated but not resampled. Texture and other appearance information of the new surface are approximated from the zombies.

8.2 Discussion

During our research with Smoothed Particle Hydrodynamics and point-based representations we have become acquainted with the advantages, drawbacks and limitations of these meshless approaches in the context of physics-based animations. Generally, we can state that meshless methods are very versatile and most suitable for animations with strong deformations or topological changes. By combining particle-based methods for volume simulation and a point-sampled surface, new effects can be achieved such as the melting of solid objects with textured and geometrically detailed surfaces to fluids. We will next discuss our findings for SPH and point-based representations in more detail.

8.2.1 Smoothed Particle Hydrodynamics

The strength of meshless methods (MMs) is their *adaptivity*. The applied numerical methods to solve the PDEs do not depend on the distribution of interpolation nodes and their connectivity. This makes MMs especially suitable for strong deformations and topological changes. Furthermore, the volume is discretized by simple sampling the domain, whereas mesh generation for the domain is problematic for both Eulerian and mesh-based Lagrangian methods. In the former the mesh has to align with complex deformable boundaries, whereas in mesh-based Lagrangian methods such as FEM the treatment of large deformations requires special remeshing techniques that are tedious and time consuming. The adaptivity of the shape functions of MMs to the distribution of the interpolation points comes at the price of higher computational costs for computing them. MMs that solve the PDEs in the *strong form* (such as collocation methods) have the advantage that they are simple to implement and efficient. However, they are often less stable and less accurate than *weak form* methods (such as the *Galerkin method* [LM54]), especially when dealing with Neumann boundary conditions.

SPH is a simple and efficient meshless collocation method. Unlike other MMs, the particles are not just used as interpolation points but carry material properties, thus SPH combines the advantages of meshless and Lagrangian methods. Therefore, material quantities such as mass are guaranteed to be preserved, in contrast to Eulerian methods where mass dissipation is a common problem. Furthermore, material properties can change as a function of time, position and neighboring particles, which can be used, for instance, to visualize the mixture of two differently colored fluids, or to simulate local melting and freezing as shown in this dissertation. Interacting with particles is very simple because external forces can be applied directly onto the particles. Furthermore, particles are suitable for controlling simulations as we have shown in [TKPR06], and hard constraints can be easily enforced by simply displacing particles, which is often useful in computer graphics applications.

We found that particle-based methods are very suitable for dynamically adapting the particle resolution. Generating new particles or merging particles is simple, and shape functions automatically adapt to the changed sampling. Similarly, discontinuities in the physical domain can be easily introduced with no need to align the discontinuities to a mesh or to remesh the domain.

A major advantage of SPH is that differentiating the field functions is simply done by differentiating the smoothing kernel, thus, no equation system needs to be solved. Hence, the gradient terms of PDEs can be written in terms of the properties of the particles. However, the simplicity of SPH comes at the price that the approximation does not fulfill the *partition of unity* property. This makes numerical analysis of the method very challenging. So far, accuracy and convergence properties of the SPH method are, to the best of our knowledge, only done for uniformly distributed particles or for certain idealized scenarios, and often only for one-dimensional cases [Mon82, SHA95, MV96]. Gingold and Monaghan [GM78] state that the accuracy of SPH is in practice much better than predicted from interpolation errors due to the energy preserving properties of the equation of motion. However, it seems hard to argue why SPH actually works that well. Other methods that are at least zeroth order consistent, such as the Moving Least-Squares Particle Hydrodynamics method, have the disadvantage that conservation of momentum is not preserved and they are considerably slower than the standard SPH method [Mon05].

The particle deficiency and tensile instability problem discussed in Section 2.3.1 is a direct consequence of not having a partition of unity. We could improve this by mirroring particles at the boundary that are then used as ghost particles for the field function computations. However, this is only applicable if the boundary is simple, and is especially difficult to apply at the interface.

One of the major difficulties we faced with the SPH method is due to the approximation of incompressibility. Pressure forces are applied that repel particles to achieve a desired rest density (see Section 4.3). These forces can be seen as springs with different rest lengths. Thus, we get the well known oscillation problems around the rest position. By making the spring stiffer (higher k^{gas}) the fluid

is less compressible but requires smaller time steps. This is a problem in computer graphics where often big time steps are used and stability is a major requirement. If we reduce the spring stiffness, the oscillations are visually very disturbing. However, making the velocity field divergence free requires solving a global linear system, which is quite expensive and we loose the nice property that all computations are performed locally.

A related problem is that the simulated fluids appear quite viscous. This is due to the slow propagation of a local change in the particle positions to the rest of the fluid. We believe that enforcing incompressibility yields less viscous fluids because a local change would result in an immediate global adaption of the particles.

Eulerian fluid simulation in combination with (particle) level sets are capable to simulate smooth, thin water sheets due to the smoothing inherent in the level set method. Particles align naturally in a hexagonal grid to minimize the internal energy. To counteract this at the interface, surface tension forces are applied for particles close to the interface. However, these surface tension forces tend to cluster small groups of particles rather than aligning them in a sheet. This makes it very hard to model thin water sheets.

To conclude, SPH is due to its adaptive nature especially suited for strong deformations and topological changes where the spatial discretization needs to be adapted, or when discontinuities are introduced into the domain. Furthermore, we believe that due to its versatility, the SPH method is suitable for combining solid and fluid animations for simulating their interaction including melting and freezing. As discussed above, there are several problems that have to be solved so that SPH fluid simulation can compete with state of the art fluid solver. However, SPH is a comparably new technique that gained increasing attention recently, especially also in the CFD community. Thus, we are confident that the SPH method will emerge as an interesting particle-based alternative to standard techniques such as the Finite Difference and Finite Element methods for solid and fluid simulation.

8.2.2 Point-based Representation

In this dissertation a point-based representation has been used for animating a surface that is embedded into the particles. This yields similar advantages as with particle-based volumetric representations, namely that the surface points (surfels) can be moved or resampled without remeshing. However, one has to be aware that volumetric remeshing, e.g., of tetrahedra, is much more involved than remeshing of a surface. Especially, distorted volumetric meshes can result in drastic degradation of accuracy or even end the computation altogether, whereas bad aspect ratio surface triangles are not very problematic for rendering. However, in case where the surface changes its topology frequently, maintaining the mesh connectivity is a computational burden that can be avoided by using a point-based representation. Another advantage of our point-based representation is the underlying im-

plicit surface definition that gives access to efficient inside/outside tests, distance to surface computations and projection of points. A major disadvantage of point-sampled surfaces is that sharp features have to be modeled explicitly. However, this can be done efficiently at render time as discussed in Section 5.6.3.

A problem of our resampling algorithm is that there is no guarantee that the whole surface is covered with surfels. Thus, the number of iterations where the surface forces are applied has to be chosen conservatively. Furthermore, for highly splashing fluids the number of surfels increases drastically, which slows down the surface animation.

We showed that point-sampled surfaces are suitable for combining the modeling of surfaces with geometric details and implicit handling of topological changes. This is achieved by blending between the detailed explicit representation and the implicit representation defined by the particles according to a scalar value. This works well because this value also indicates if topological changes are likely to occur. However, as long as the surface does not fully correspond to the implicit representation a consistent surface without self-intersections is not guaranteed.

To conclude, surfels are a versatile surface primitive suitable for embedding a surface into the volumetric particles. This meshless representation is very efficient for strongly deforming surfaces or surfaces that change their topology frequently during a simulation because no mesh needs to be maintained. The combination of explicit and implicit representation of the surface defined by the surfels is very powerful, and can be further extended to adapt to the isosurface defined by the underlying particles.

8.3 Future Work

Research of physics-based meshless Lagrangian animation, where both the volume and the surface are point-based, has only just started. Thus, many open problems remain in this interesting area.

As discussed above, one of the major shortcomings of SPH is its approximation of incompressibility. For strictly enforcing incompressibility, and thus guaranteeing a divergence free velocity field, a Poisson equation for the pressure has to be solved. Approaches exist to solve this equation directly on the particles [KTO96, CEL06]. This yields a $n \times n$ sparse linear equation system with n the number of particles. Thus, with hundreds of thousands of particles, solving this system becomes intractable. Another possibility is to use a similar approach to the Fluid-Implicit-Particle (FLIP) method [BR86, ZB05], where a staggered (MAC) mesh (or grid) is used for pressure projection. The face velocities are approximated from the particle velocities, for instance, using MLS. The new velocity is then computed, and the difference between the new and old velocity is interpolated back onto the particles. Using a mesh has the advantage that the grid resolution is independent of the particle resolution. Because the mesh is only used

to compute a divergence free velocity field, a different mesh can be used in every time step. Similar to Klingner et al. [KFCO06], an unstructured mesh could be used that aligns to the boundary, where the resolution of the mesh could adapt to the particles. Furthermore, each separated fluid part can be handled independently, thus keeping the advantage that the simulation is not restricted to a certain area in space.

For computing accurate solutions with SPH, sufficient overlap of the smoothed particles must be guaranteed. Furthermore, the approximation of field functions is sensitive to particle disorder, especially for higher derivatives. To remedy these problems, Chaniotis et al. [CPK02] suggest to reinitialize the particle locations periodically onto a uniform grid by interpolating the particle properties using high order interpolation kernels that conserve the total momentum. However, the resampling introduces numerical errors and extra diffusion that can be quite large as pointed out by the authors. Nevertheless, this approach could be interesting, for instance, in connection with solving for incompressibility as described before.

We use a two-pass algorithm for computing the SPH forces. First, the density is computed (Equation (4.5)), which is then used to approximate the field variables. This requires storing the neighbors of a particles, which can become a bottleneck with large number of particles. SPH forces can be computed together with the density if the continuity equation (Equation (4.2)) is solved directly, however, in our experiments this yielded instabilities due to the large time steps that are used for our animations. Thus, we want to look for different approaches to compute the density.

Besides the surface, the bottleneck of our physics simulation for large numbers of particles is the neighborhood computations. Therefore, new search data structures and neighbor caching schemes need to be developed that exploit the temporal coherence of neighborhoods.

Because all computations in the SPH method are local, they can be easily parallelized on a cluster [GJ01] or the GPU [HCM06], including the neighborhood queries. For parallel computations, a one-pass SPH algorithm would yield even stronger performance gains than for the CPU.

The two-way coupling between air and water particles described in Section 4.4 requires a band of air particles sampled around the water interface. This drastically increases the number of particles that have to be animated, especially because these particles are all on the highest resolution. Instead of a full simulation, the air particles could be simply advected. However, ensuring a sufficient sampling around the interface without introducing a discontinuity in the surface computation due to changes in the color field is challenging. Furthermore, a consistent transition from advected air to simulated bubbles is needed.

The bottleneck in the surface animation is the repulsion and resampling scheme, which is applied to ensure a hole free surface and a locally uniform distribution. An improved sampling scheme would change the particle position only where necessary and can guarantee a hole free surface sampling.

A problem of the implicit surface definition from the particles (Section 2.5.1) is

that the applied Euclidian distance function is not adapted to topology, which can result in a bad approximation of the point-sampled surface with even wrong topology. To prevent this, the geodesic distance of the neighbors has to be considered instead of the Euclidian distance. Klein and Zachmann [KZ04] build a proximity neighbor graph either from a Delaunay or sphere-of-influence graph. Efficiently maintaining this graph during a simulation is an interesting task for future work.

Detection of self-intersections for deforming objects is a hard problem, especially for point-sampled surfaces because the surfels overlap by definition. Image-based techniques have been used to detect self-collisions of triangle meshes [BW02, HTG04]. Similarly, image-based methods in combination with splatting on the GPU might be exploited for our purposes.

As already discussed in the introduction, in computer graphics it is often more important that an animation looks realistic than that it is physically accurate. Thus, we believe that user studies are needed to find out when an animation looks realistic and where the deficiencies are. Furthermore, animation are often only useful if they can be controlled by the designer. For instance in fluid animations, she might want to control the fluid flow, position and strength of vortices, the amount of splashes, the interaction with objects, the "fluid shape", bubbles, hard constraints for where the fluid (including splashes) is allowed to enter, and so on. The control can take place on different scales. For instance, the user might only want to control the fluid flow on a low scale and preserve the fine scale detail, or she might want to change the high frequency behavior of the fluid to generate more splashes. In [TKPR06] we have shown that particles are suitable for these kind of controls, in the future we want to explore additional control possibilities. To make control more intuitive, sketch-based techniques could be applied. Sketch-based interfaces have already been applied successfully for free-form shape modeling [IMT99, KH02, DE03, NSACO05, KH06]. Similarly, a simulation could be designed and controlled with simple and intuitive drawing operations.

Bibliography

[AA03a] Anders Adamson and Marc Alexa. Approximating and intersecting surfaces from points. In *SGP '03: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 230–239. Eurographics Association, 2003.

[AA03b] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In *Proceedings of Shape Modeling International*, pages 272–279, 2003.

[AA04a] Anders Adamson and Marc Alexa. Approximating bounded, non-orientable surfaces from points. In *Proceedings of Shape Modeling International 2004*. IEEE Computer Society, 2004. accepted for publication.

[AA04b] Anders Adamson and Marc Alexa. On normals and projection operators for surfaces defined by point sets. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Point-Based Graphics*, pages 149–156. Eurographics Association, 2004.

[ABCO+01] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Cláudio T. Silva. Point set surfaces. In *Proceedings of the conference on Visualization '01*, 2001.

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Cláudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Computer Graphics and Visualization*, 9(1):3–15, 2003.

[AD03] Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings)*, 22(3):651–656, July 2003.

[Ada06] Bart Adams. *Point-based Modeling, Animation and Rendering of Dynamic Objects*. PhD thesis, Katholieke Universiteit Leuven, Belgium, May 2006.

[AF00] Sunil Arya and Ho-Yam A. Fu. Expected-case complexity of approximate nearest neighbor searching. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 379–388. Society for Industrial and Applied Mathematics, 2000.

[AK04] Nina Amenta and Yong Kil. Defining point-set surfaces. *ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings)*, 23(3):264–270, 2004.

[AKP+05] Bart Adams, Richard Keiser, Mark Pauly, Leonidas J. Guibas, Markus Gross, and Philip Dutré. Efficient raytracing of deforming point-sampled surfaces. *Computer Graphics Forum (Eurographics 2005 Proceedings)*, 24(3):677–684, 2005.

[AM93] S. Arya and D.M. Mount. Algorithms for fast vector quantization. In *Data Compression Conference*, pages 381–390. IEEE Computer Society Press, 1993.

[And95] Ted L. Anderson. *Fracture Mechanics*. CRC Press, 1995.

[AWD+04] Bart Adams, Martin Wicke, Philip Dutré, Markus Gross, Mark Pauly, and Matthias Teschner. Interactive 3D painting on point-sampled objects. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 57–66, 2004.

[Bal95] Dinshaw S. Balsara. von neumann stability analysis of smoothed particle hydrodynamics - suggestions for optimal algorithms. *Journal of Computational Physics*, 121(2):357–372, 1995.

[Bau75] Bruce G. Baumgart. A polyhedron representation for computer vision. In *Proceedings of National Computer Conference*, volume 44, pages 589–596, 1975.

[Ben75] Jon L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Ben90] W. Benz. Smooth particle hydrodynamics: a review. *The Numerical Modelling of Nonlinear Stellar Pulsation*, pages 269–288, 1990.

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 21(3):594–603, August 2002.

[BGOS06] A.W. Bargteil, T.G. Goktekin, J.F. O'Brien, and J.A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1), 2006.

[BGTG03] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graphics*, pages 377–386, 2003.

[BHW94] D. Breen, D. House, and M. Wozny. Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH '94*, pages 365–372, New York, NY, USA, 1994. ACM Press.

[BHZK05] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's GPUs. In *Proceedings of Symposium on Point-Based Graphics 2005*, pages 17–24, 2005.

[BK00] J. Bonet and S. Kulasegaram. Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations. *International Journal for Numerical Methods in Engineering*, 47(6):1189–1214, 2000.

[BK03a] Stephan Bischoff and Leif Kobbelt. Parameterization free active contour models with topology control. In *4th Isreal-Korean Binational Conference on Geometric Modeling and Computer Graphics*, pages 69–74, 2003.

[BK03b] Stephan Bischoff and Leif Kobbelt. Sub-voxel topology control for level-set surfaces. *Computer Graphics Forum*, 22(3):273–280, September 2003.

[BLG94] T. Belytschko, Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[BLM00] Ted Belytschko, Wing Kam Liu, and Brian Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons Ltd., 2000.

[BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH/Eurographics Symposium Computer Animation*, pages 28–36. ACM Press, 2003.

[BO84] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53, 1984.

[BOT01] S. Borve, M. Omang, and J. Trulsen. Regularized smoothed particle hydrodynamics: A new approach to simulating magnetohydrodynamic shocks. *The Astrophysical Journal*, 561(1):82–93, November 2001.

[BOT05] S. Borve, M. Omang, and J. Trulsen. Regularized smoothed particle hydrodynamics with improved multi-resolution handling. *Journal of Computational Physics*, 208(1):345–367, 2005.

[BR86] J.U. Brackbill and H.M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.

[BSK04] Mario Botsch, Michael Spernat, and Leif Kobbelt. Phong splatting. In *Proceedings of Symposium on Point-Based Graphics 2004*, pages 25–32, 2004.

[BSSH04] G. Bianchi, B. Solenthaler, G. Székely, and M. Harders. Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations. In *MICCAI (2)*, pages 293–301, 2004.

[BTH+03] Kiran Bhat, Christopher Twigg, Jessica K. Hodgins, Pradeep Khosla, Zoran Popovic, and Steven Seitz. Estimating cloth simulation parameters from video. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 2003.

[BW92] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 303–308. ACM Press, 1992.

[BW97] J. Bonet and R.D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge Univ. Press, NY, 1997.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54. ACM, ACM Press / ACM SIGGRAPH, 1998.

[BW02] George Baciu and Wingo Sai-Keung Wong. Hardware-assisted self-collision for deformable surfaces. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 129–136, New York, NY, USA, 2002. ACM Press.

[BYM05] Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 77–86, New York, NY, USA, 2005. ACM Press.

[Can93a] Marie-Paule Cani. An implicit formulation for precise contact modeling between flexible solids. In *SIGGRAPH '93*, pages 313–320, 1993.

[Can93b] Marie-Paule Cani. An implicit formulation for precise contact modeling between flexible solids. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 313–320. ACM Press, 1993.

[Car04] Mark T. Carlson. *Rigid, Melting, and Flowing Fluid*. PhD thesis, Georgia Institute of Technology, 2004.

[CBC99] J.K. Chen, J.E. Beraun, and T.C. Carney. A corrective smoothed particle method for boundary value problems in heat conduction. *International Journal for Numerical Methods in Engineering*, 46(2):231–252, 1999.

[CBJ99a] J.K. Chen, J.E. Beraun, and C.J. Jih. Completeness of corrective smoothed particle method for linear elastodynamics. *Computational Mechanics*, 24:273–285, 1999.

[CBJ99b] J.K. Chen, J.E. Beraun, and C.J. Jih. An improvement for tensile instability in smoothed particle hydrodynamics. *Computational Mechanics*, 23(4):279–287, 1999.

[CBP05] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005*, pages 219–228, July 2005.

[CDZC01] Philippe Chanzy, Luc Devroye, and Carlos Zamora-Cura. Analysis of range search for random k-d trees. *Acta Informatica*, 37(4/5):355–383, 2001.

[CEL06] F. Colin, R. Egli, and F.Y. Lin. Computing a null divergence velocity field using smoothed particle hydrodynamics. *Journal of Computational Physics*, 2006. to be published.

[CFL28] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[CGFO06] Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, 2006. To appear.

[CH02] Liviu Coconu and Hans-Christian Hege. Hardware-oriented point-based rendering of complex scenes. In *Proceedings Eurographics Workshop on Rendering*, pages 43–52, 2002.

[CHP89] J. Chadwick, D. Haumann, and R. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89*, pages 243–252, New York, NY, USA, 1989. ACM Press.

[Chu96] T.J. Chung. *Applied Continuum Mechanics*. Cambridge Univ. Press, NY, 1996.

[CK02] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 604–611, New York, NY, USA, 2002. ACM Press.

[CL03] Andrea Colagrossi and Maurizio Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191(2):448–475, 2003.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[CMPW89] Robert D. Cook, David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, New York, 1989.

[CMT04] Mark Carlson, Peter John Mucha, and Greg Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transations on Graphics*, 23(3):377–384, August 2004.

[CMVT02] Mark Carlson, Peter Mucha, Brooks Van Horn III, and Greg Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation*. ACM Press / ACM SIGGRAPH, 2002.

[Coh91] Laurent D. Cohen. On active contour models and balloons. *CVGIP: Image Underst.*, 53(2):211–218, 1991.

[Coo95] R.D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, NY, 1995.

[CPK02] A.K. Chaniotis, D. Poulikakos, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *Journal of Computational Physics*, 182(1):67–90, 2002.

[CR99] Sharen J. Cummins and Murray Rudman. An sph projection method. *Journal of Computational Physics*, 152, 1999.

[DC94] Mathieu Desbrun and Marie-Paule Cani. Highly deformable material for animation and collision processing. In *Fifth Eurographics Workshop on Animation, Simulation*, September 1994.

[DC95] Mathieu Desbrun and Marie-Paule Cani. Animating soft substances with implicit surfaces. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, volume 29 of *Annual Conference Series*, pages 287–290. ACM SIGGRAPH, Addison Wesley, aug 1995. Los Angeles, California, published under the name Marie-Paule Gascuel.

[DC96] Mathieu Desbrun and Marie-Paule Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96*, pages 61–76, 1996.

[DC98] Mathieu Desbrun and Marie-Paule Cani. Active implicit surface for animation. In *Proceedings of Graphics Interface*, pages 143–150, 1998.

[DC99] Mathieu Desbrun and Marie-Paule Cani. Space-time adaptive simulation of highly deformable substances. Technical report, INRIA Nr. 3829, 1999.

[DCA99] Hervé Delingette, Stéphane Cotin, and Nicholas Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *CA '99: Proceedings of the Computer Animation*, page 70, Washington, DC, USA, 1999. IEEE Computer Society.

[DDBC99] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation '99*, pages 133–144, 1999.

[DDCB00] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Adaptive simulation of soft bodies in real-time. In *Computer Animation '00*, pages 15–20, 2000.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series, pages 31–36. ACM SIGGRAPH 2001, August 2001.

[DE03] Geoffrey M. Draper and Parris K. Egbert. A gestural interface to free-form deformation. In *Graphics Interface*, pages 113–120. CIPS, Canadian Human-Computer Commnication Society, A K Peters, June 2003. ISBN 1-56881-207-8, ISSN 0713-5424.

[DH04] Jean Donea and Antonio Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, December 2004.

[Dil99a] Gary A. Dilts. Moving-least-squares-particle hydrodynamicsă–ăi. consistency and stability. *International Journal for Numerical Methods in Engineering*, 44(8):1115–1155, 1999.

[Dil99b] Gary A. Dilts. Moving-least-squares-particle hydrodynamicsă-ăi. consistency and stability. *International Journal for Numerical Methods in Engineering*, 44(8):1115–1155, 1999.

[DRI97] C.T. Dyka, P.W. Randles, and R.P. Ingel. Stress points for tension instability in sph. *International Journal for Numerical Methods in Engineering*, 40(13):2325–2341, 1997.

[DSB99] Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Interactive animation of structured deformable objects. In *Graphics Interface '99*, 1999.

[DVS03] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. In *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings)*, pages 657–662. ACM Press, 2003.

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[EGS03] Olaf Etzmuss, Joachim Gross, and Wolfgang Strasser. Deriving a particle system from continuum mechanics for the animation of deformable objects.

IEEE Transactions on Visualization and Computer Graphics, 9(4):538–550, 2003.

[ELF05a] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set. *Computers and Structures*, 83:479–490, 2005.

[ELF05b] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. In *Computers and Structures*, volume 83, pages 479–490, 2005.

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *ACM Transactions on Graphics (SIGGRAPH 2002 Proceedings)*, pages 736–744, 2002.

[ETK+05] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun. Discrete, circulation-preserving, and stable simplicial fluids. Technical report, Caltech, 2005.

[EWS96] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, 1996.

[FBF77] Jerome H. Freidman, Jon L. Bentley, and Raphael A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[FCOAS03] Shachar Fleishman, Daniel Cohen-Or, Marc Alexa, and Cláudio T. Silva. Progressive point set surfaces. *ACM Transactions on Graphics*, 22(4):997–1011, 2003.

[FCOS05] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3):544–552, 2005.

[FdH+87] Uriel Frisch, Dominique d'Humieres, Brosl Hasslacher, Pierre Lallemand, Yves Pomeau, and Jean-Pierre Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.

[Fed02] Ronald P. Fedkiw. Coupling an eulerian fluid calculation to a lagrangian solid calculation with the ghost fluid method. *Journal on Computational Physics*, 175(1):200–224, 2002.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 23–30, 2001.

[FGTV92] Luiz Henrique de Figueiredo, Jonas de Miranda Gomes, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of the conference on Graphics interface '92*, pages 250–257, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[FLS63] R.P. Feynman, R.B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 1. Addison-Wesley, 1963.

[FM96] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[FM97a] Nick Foster and Dimitri Metaxas. Controlling fluid animation. In *Proceedings of CGI '97*, pages 178–188, 1997. Winner of the Androme Award 1997.

[FM97b] Nick Foster and Dimitri Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97*, pages 181–188, 1997.

[FM03] Thomas-Peter Fries and Hermann G. Matthies. Classification and overview of meshfree methods. Technical report, TU Brunswick, Germany Nr. 2003-03, 2003.

[FMH04] S. Fernandez-Mendez and A. Huerta. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering*, 193:1257–1275, 2004.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3), 2005.

[FOKG05] Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. Fluids in deforming meshes. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 255–259, New York, NY, USA, 2005. ACM Press.

[FPRJ00] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 249–254. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[FR86] Alain Fournier and William T. Reeves. A simple model of ocean waves. In *SIGGRAPH '86*, pages 75–84, New York, NY, USA, 1986. ACM Press.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01*, pages 15–22, 2001.

[Fun94] Y.C. Fung. *A First Course in Continuum Mechanics*. Prentice-Hall, Englewood Cliffs, N.J., third edition, 1994.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings)*, 23:463–468, 2004.

[GH04] S.T. Greenwood and D.H. House. Better with bubbles: enhancing the visual realism of simulated fluid. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 287–296, New York, NY, USA, 2004.

[GHD03] Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. Simulating fluid-solid interaction. In *Graphics Interface*, pages 31–38. CIPS, Canadian Human-Computer Commnication Society, A K Peters, June 2003. ISBN 1-56881-207-8, ISSN 0713-5424.

[GHDS03] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 62–67. Eurographics Association, 2003.

[GJ01] R.J. Goozee and P.A. Jacobs. Distributed and shared memory parallelism with a smoothed particle hydrodynamics code. In *Proceedings of CTAC 2001*, 2001.

[GKS02] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A simple framework for adaptive simulation. In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 281–290. ACM, ACM Press / ACM SIGGRAPH, 2002.

[GM77] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices*, 181:375–389, 1977.

[GM78] R.A. Gingold and J.J. Monaghan. Binary fission in damped rotating polytropes. *Monthly Notices of the Royal Astronomical Society*, 184:481–499, 1978.

[GM95] Leonidas J. Guibas and David H. Marimont. Rounding arrangements dynamically. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 190–199. ACM Press, 1995.

[GMS01] J.P. Gray, J.J. Monaghan, and R.P. Swift. Sph elastic dynamics. *Computer Methods in Applied Mechanics and Engineering*, 190(49):6641–6662, 2001.

[Gos90] Michael E. Goss. Motion simulation: A real time particle system for display of ship wakes. *IEEE Comput. Graph. Appl.*, 10(3):30–35, 1990.

[GP06] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. 2006. to appear.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3):973–981, 2005.

[Hah88] J.K. Hahn. Realistic animation of rigid bodies. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 299–308. ACM Press, 1988.

[Har63] F.H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. *Experimental arithmetic, high-speed computations and mathematics*, 1963.

[HCM06] Kyle Hegeman, Nathan A. Carr, and Gavin S.P. Miller. Particle-based fluid simulation on the gpu. In Vassil N. Alexandrov, Geert D. van Albada, Peter M.A. Sloot, and Jack Dongarra, editors, *Computational Science – ICCS 2006*, volume 3994 of *LNCS*, pages 228–235. Springer, 2006.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):71–78, July 1992.

[Hec97] Paul S. Heckbert. Fast surface particle repulsion. Technical report, CMU Computer Science, 1997.

[HES03] Michael Hauth, Olaf Etzmuss, and Wolgang Straßer. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19:581–600, December 2003.

[HK03] Jeong-Mo Hong and Chang-Hun Kim. Animation of bubbles in liquid. *Computer Graphics Forum*, 22(3), 2003.

[HK05a] Simone E. Hieber and Petros Koumoutsakos. A lagrangian particle level set method. *Journal of Computational Physics*, 210(1):342–367, nov 2005.

[HK05b] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3):915–920, 2005.

[HMT01] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as continuum. *Computer Graphics Forum (Proceedings of Eurographics 2001)*, 20(3), 2001.

[HN81] C.W. Hirt and B.D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[HNB+05] Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. Gigantic deformable surfaces. In *Proceedings of the SIGGRAPH 2005 on Sketches & Applications*, New York, NY, USA, 2005. ACM Press.

[HNB+06] Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, 25(1):151–175, 2006.

[Hop94] Hugues Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Computer Science and Engineering, University of Washington, June 1994.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM Press.

[HPH96] D. Hutchinson, M. Preston, and T. Hewitt. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 31–45, New York, NY, USA, 1996. Springer-Verlag New York, Inc.

[HSIW96] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles: Range image fusion for complex object modeling. In *International Conference on Image Processing*, 1996.

[HTG04] Bruno Heidelberger, Matthias Teschner, and Markus Gross. Detection of collisions and self-collisions using image-space techniques. *Journal of WSCG*, 12:145–152, 2004.

[HTK98] Koichi Hirota, Yasuyuki Tanoue, and Toyohisa Kaneko. Generation of crack patterns with a physical model. *The Visual Computer*, 14:126–137, 1998.

[HTK+04] Bruno Heidelberger, Matthias Teschner, Richard Keiser, Matthias Müller, and Markus Gross. Consistent penetration depth estimation for deformable collision response. In *Proceedings of Vision, Modeling, Visualization VMV'04*, pages 339–346, 2004.

[Hun05] Peter Hunter. *FEM/BEM Notes*. University of Oakland, New Zealand, 2005. http://www.bioeng.auckland.ac.nz/cmiss/fembemnotes/fembemnotes.pdf.

[HWB04] Ben Houston, Mark Wiebe, and Christopher Batty. Rle sparse level sets. In *Proceedings of the SIGGRAPH 2004 on Sketches & Applications*, New York, NY, USA, 2004. ACM Press.

[HXP01] X. Han, C. Xu, and J. Prince. A topology preserving deformable model using level sets. In *Computer Vision and Pattern Recognition Proceedings 01*, pages 765–770, 2001.

[IGLF06] Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, 25(3):805–811, 2006.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[ITF04] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 131–140. ACM Press, 2004.

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):339–346, 2002.

[JP99] Doug L. James and Dinesh K. Pai. ArtDefo: accurate real time deformable objects. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[JP02] Doug L. James and Dinesh K. Pai. Real time simulation of multizone elastokinematic models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 927–932, 2002.

[JP03] Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22(1):47–82, 2003.

[JP04] Doug L. James and Dinesh K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings)*, 23(3):393–398, 2004.

[KAG+05] Richard Keiser, Bart Adams, Dominique Gasser, Paolo Bazzi, Philip Dutré, and Markus Gross. A unified lagrangian approach to solid-fluid animation. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 125–148, 2005.

[KAG+06] Richard Keiser, Bart Adams, Leonidas J. Guibas, Philip Dutré, and Mark Pauly. Multiresolution particle-based fluids. CS Technical Report 520, Computer Science Department, ETH Zurich, Switzerland, 2006.

[Kal89] Y.E. Kalay. The hybrid edge: a topological data structure for vertically integrated geometric modelling. *Computer Aided Design*, 21(3):130–140, 1989.

[Kau87] A. Kaufman. Efficient algorithms for 3d scanconversion of parametric curves, surfaces, and volumes. In *Computer Graphics*, SIGGRAPH 87 Proceedings, pages 171–179, 1987.

[KB03] Leif Kobbelt and Mario Botsch. Freeform shape representations for efficient geometry processing. *smi*, 00:111, 2003.

[KBLRP00] S. Kulasegaram, J. Bonet, T.-S.L. Lok, and M. Rodriguez-Paz. Corrected smooth particle hydrodynamics - a meshless method for compoutational mechanics. Technical report, EC-COMAS, 2000.

[KBSS01] Leif Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 57–66, 2001.

[KCC+06] Janghee Kim, Deukhyun Cha, Byungjoon Chang, Bonki Koo, and Insung Ihm. Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, 2006. To appear.

[Kei03] Richard Keiser. Collision detection and response for interactive editing of point-sampled models. Master's thesis, ETH, Zürich, Switzerland, 2003.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Transactions on Graphics (SIGGRAPH 2006 Proceedings)*, August 2006. to appear.

[KH02] Olga Karpenko and John F. Hughes. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3), 2002.

[KH06] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics*, 25(3):589–598, 2006.

[KLLR05] ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. Flowfixer: Using bfecc for fluid simulation. *Eurographics Workshop on Natural Phenomena*, pages 51–56, 2005.

[KM90] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 49–57, New York, NY, USA, 1990. ACM Press.

[KMH+04] Richard Keiser, Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Contact handling for deformable point-based objects. In *Proceedings of Vision, Modeling, Visualization (VMV) '04*, pages 339–347, November 2004.

[Kou05] Petros Koumoutsakos. Multiscale flow simulations using particles. *Annu. Rev. Fluid Mech.*, 37, 2005.

[KTO95] S. Koshizuka, H. Tamako, and Y. Oka. A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Dynamics*, 4:29–46, 1995.

[KTO96] S. Koshizuka, H. Tamako, , and Y. Oka. A particle method for incompressible viscous flow with fluid fragmentation. *Journal of Computational Fluid Dynamics*, 29, 1996.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[KYT+06] Liliya Kharevych, Weiwei Yang, Yiying Tong, Mathieu Desbrun, Peter Schroeder, Eva Kanso, and Jerrold E. Marsden. Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, 2006. To appear.

[KZ04] J. Klein and G. Zachmann. Point cloud collision detection. *Computer Graphics Forum (Proceedings of EUROGRAPHICS)*, 23:567–576, 2004.

[LC87] William Lorensen and Harvey Cline. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. In *Computer Graphics Vol. 21, No. 4*, pages 163–169, August 1987.

[Lev98] David Levin. The approximation power of moving least-squares. *Math. Comput.*, 67(224):1517–1531, 1998.

[Lev01] David Levin. Mesh-independent surface interpolation. In *Advances in Computational Mathematics*, 2001.

[Lev04] David Levin. Mesh-independent surface interpolation. In Guido Brunnett, Bernd Hamann, Heinrich Müller, and Lars Linsen, editors, *Geometric Modeling for Scientific Visualization*. Springer, 2004.

[LFDL98] J. Lasenby, W.J. Fitzgerald, C.J.L. Doran, and A.N. Lasenby. New geometric methods for computer vision – an application to structure and motion estimation. *International Journal of Computer Vision*, 26(3):191–213, 1998.

[LFO05] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 2005.

[LGF04] Frank Losasso, Frederic Gibou, and Ronald Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, 2004.

[LH06] Sylvain Lefebvre and Hugues Hoppe. Perfect spatial hashing. *ACM Transactions on Graphics (SIGGRAPH 2006 Proceedings)*, August 2006. to appear.

[LIGF06] Frank Losasso, Geoffrey Irving, Eran Guendelman, and Ron Fedkiw. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):343–352, 2006.

[Lin01] Lars Linsen. Point cloud representation. Technical Report 2001-3, Faculty for Computer Science, Universität Karlsruhe, 2001.

[Liu02a] G. R. Liu. *Mesh-Free Methods*. CRC Press, 2002.

[Liu02b] I-Shih Liu. *Continuum Mechanics*. Springer-Verlag, Berlin, 2002.

[LL86] L.D. Landau and E.M. Lifshitz. *Theory of elasticity*. Pergamon Press, Oxford, 1986.

[LL87] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics: Course of Theoretical Physics*, volume 6. Butterworth-Heineman, 2nd edition, 1987.

[LL02] S. Li and W.K. Liu. Meshfree and particle methods and their applications. *Appl. Mech. Rev.*, 55:1–34, 2002.

[LL03] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics, A Meshfree Particle Method*. World Scientific Publishing, 2003.

[LM54] P.D. Lax and A.N. Milgram. Parabolic equations. *Annals of mathematics studies*, 33:167–190, 1954.

[Lov27] A.E.H. Love. *A Treatise on the Mathematical Theory of Elasticity*. Cambridge University Press, 1927.

[LPC+93] Larry D. Libersky, Albert G. Petschek, Theodore C. Carney, Jim R. Hipp, and Firooz A. Allahdadi. High strain lagrangian hydrodynamics: a three-dimensional sph code for dynamic material response. *Journal of Computational Physics*, 109(1):67–75, 1993.

[LRK93] W. Michael Lai, David Rubin, and Erhard Krempl. *Introduction to Continuum Mechanics*. Pergamon Press, Headington Hill Hall, Oxford, England, 1993.

[LSSF06] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*, 25(3):812–819, 2006.

[Luc77] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013–1024, 1977.

[LV05] Ling Li and Vasily Volkov. Cloth animation with adaptively refined meshes. In *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*, pages 107–113, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[LW77] D.T. Lee and C.K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9(1):23 – 29, March 1977.

[MBF04] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings)*, 23(3):385–392, 2004.

[MC95] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–188. ACM Press, 1995.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159. Eurographics Association, 2003.

[McM06] Steve McMillan. Computational physics course notes, 2006. [Online; accessed 3-August-2006].

[MDM+02] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–54, New York, NY, USA, 2002. ACM Press.

[MF] Emud Mokhberi and Petros Faloutsos. An implementation of a particle level set library. `http://www.magix.ucla.edu/software/levelSetLibrary/`.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of the 2004 conference on Graphics Interface*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3):471–478, 2005.

[MHW05] Joseph J. Monaghan, Herbert E. Huppert, and M. Grae Worster. Solidification using smoothed particle hydrodynamics. *Journal of Computational Physics*, 206(2):684–705, 2005.

[Mil88] Gavin S.P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH '88*, pages 169–173, New York, NY, USA, 1988. ACM Press.

[MKN+04] Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 141–151. ACM Press, 2004.

[MLM+05] U. Meiser, O. Lopez, C. Monserrat, M.C. Juan, and M. Alcaniz. Real-time deformable models for surgery simulation: a survey. *Computer Methods and Programs in Biomedicine*, 77:183–197, 2005.

[MMDJ01] Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 113–124, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[MMS04] Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. Animation and control of breaking waves. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 315–324, New York, NY, USA, 2004. ACM Press.

[Mon82] J.J. Monaghan. Why particle methods work. *SIAM Journal on Scientific and Statistical Computing*, 3:422–433, 1982.

[Mon92] J.J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. and Astrophysics*, 30:543–574, 1992.

[Mon94] J.J. Monaghan. Simulating free surface flows with SPH. *Journal on Computational Physics*, 110(2):399–406, 1994.

[Mon00] J.J. Monaghan. Sph without a tensile instability. *Journal of Computational Physics*, 159(2):290–311, 2000.

[Mon05] J.J. Monaghan. Smoothed particle hydrodynamics. *Rep. Prog. Phys.*, 68:1703–1758, 2005.

[Mor00] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33(3):333–353, 2000.

[Mou05] D. Mount. Ann programming manual, 2005.

[MP89] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

[MSHG04] Matthias Müller, Simon Schirm, Bruno Heidelberger, and Markus Gross. Interaction of fluids with deformable solids. In *Computer Animation and Virtual Worlds (CAVW)*, pages 159–171, 2004.

[MSKG05] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/EurographicsSymposium on Computer Animation*, pages 237–244, New York, NY, USA, 2005. ACM Press.

[MTG04] Matthias Müller, Matthias Teschner, and Markus Gross. Physically-based simulation of objects represented by surface meshes. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 26–33, Washington, DC, USA, 2004. IEEE Computer Society.

[MV96] B.B. Moussa and J.P. Vila. Convergence of sph method for scalar nonlinear conservation laws. *SIAM Journal on Applied Mathemathics*, 37:863–887, 1996.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298. ACM Press, 1988.

[NFJ02] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728. ACM Press, 2002.

[NMK+06] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxermann, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4), 2006. To appear.

[NP00] S. Nugent and H.A. Posch. Liquid drops and surface tension with smoothed particle applied mechanics. *Phys. Rev. E*, 62(4):4968–4975, October 2000.

[NSACO05] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3):1142–1147, 2005.

[OBH02a] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics (SIGGRAPH 2002 Proceedings)*, pages 291–294, 2002.

[OBH02b] J.F. O'Brien, A.W. Bargteil, and J.K. Hodgins. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 291–294. ACM, ACM Press / ACM SIGGRAPH, 2002.

[OC97] Agata Opalach and Marie-Paule Cani. Local deformations for animation of implicit surfaces. In Wolfgang Strasser, editor, *13th Spring Conference on Computer Graphics*, pages 85–92, 1997.

[OF02] J.S. Osher and R.P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[OF03] S. Osher and R. Fedkiw. *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[OFL01] D. O'Brien, S. Fisher, and M.C. Lin. Automatic simplification of particle system dynamics. In *Computer Animation 2001*, 2001.

[OFTB96] D. Organ, M. Fleming, T. Terry, and T. Belytschko. Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Computational Mechanics*, 18:1–11, 1996.

[OH95] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation*, page 198, 1995.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Computer Graphics*, SIGGRAPH 99 Proceedings, pages 287–296. ACM, ACM Press / ACM SIGGRAPH, 1999.

[OP99] M. Ortiz and A. Pandolfi. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. *International Journal for Numerical Methods in Engineering*, 44:1267–1282, 1999.

[OS88] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[Paj05] Renato Pajarola. Stream-processing points. In *Proceedings IEEE Visualization*, pages 239–246. Computer Society Press, 2005.

[Pau03] Mark Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, Department of Computer Science, ETH Zürich, 2003.

[PB81] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *SIGGRAPH '81*, pages 245–252, New York, NY, USA, 1981. ACM Press.

[Pea86] Darwyn R. Peachey. Modeling waves and surf. In *SIGGRAPH '86*, pages 65–74, New York, NY, USA, 1986. ACM Press.

[PG01] Mark Pauly and Markus Gross. Spectral processing of point-sampled geometry. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 379–386. ACM Press, 2001.

[PGK02] Mark Pauly, Markus Gross, and Leif Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, pages 163–170, 2002.

[PKA+05] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3):957–964, 2005.

[PKG03] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum (Eurographics 2003 Proceedings)*, 22(3):281–290, 2003.

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings)*, 22(3):641–650, 2003.

[PM85] G.J. Phillips and J.J. Monaghan. A numerical method for three-dimensional simulations of collapsing, isothermal, magnetic gas clouds. *Monthly Notices of the Royal Astronomical Society*, 216:883–895, October 1985.

[PPG04] Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 109–119. ACM Press, 2004.

[PTB+03] Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. In *Proceedings of Eurographics 2003*, pages 401–410, 2003.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[Ree83] W.T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983.

[REN+04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2004.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87*, pages 25–34, New York, NY, USA, 1987. ACM Press.

[Rhe98] W. Rheinboldt. Methods for solving systems of nonlinear equations. In *CBMS-NSF regional conference series in applied mathematics*, volume 70, 1998.

[Rob81] John T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *SIGMOD Conference*, pages 10–18, 1981.

[RS72] Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics I: Functional Analysis*. Academic Press, London, San Diego, 1972.

[SAB+99] M. Sussman, A. Almgren, J Bell, P. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148, 1999.

[SAC+99] Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. Animating lava flows. In *Proceedings of Graphics Interface*, pages 203–210, 1999.

[Sam89] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, Massachusetts, 1989.

[Sam05] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2005. ISBN 0-12-369446-9.

[Set96] James A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Science*, pages 1591–1595, 1996.

[Set99] James A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[SF95] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95*, pages 129–136, New York, NY, USA, 1995. ACM Press.

[SHA95] J.W. Swegle, D.L. Hicks, and S.W. Attaway. Smoothed particle hydrodynamics stability analysis. *Journal of Computational Physics*, 116(1):123–134, 1995.

[Sig06] Christian Sigg. *Representation and Rendering of Implicit Surfaces*. PhD thesis, Federal Institute of Technology (ETH) of Zurich, March 2006.

[Sim90] Karl Sims. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90*, pages 405–413, New York, NY, USA, 1990. ACM Press.

[SOG06] Denis Steinemann, Miguel A. Otaduy, and Markus Gross. Fast arbitrary splitting of deforming objects. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006. To appear.

[SP00] M. Sussman and E.G. Puckett. A coupled level set and volume of fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162:301–337, 2000.

[SPG03] Christian Sigg, Ronald Peikert, and Markus Gross. Signed distance transform using graphics hardware. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 12, Washington, DC, USA, 2003. IEEE Computer Society.

[SRF05] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings)*, 24(3):910–914, 2005.

[SSK05] Oh-Young Song, Hyuncheol Shin, and Hyeong-Seok Ko. Stable but nondissipative water. *ACM Transactions on Graphics*, 24(1):81–97, 2005.

[ST92] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, 1992.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH '99*, pages 121–128, 1999.

[Str99] John Strain. Semi-lagrangian methods for level set equations. *J. Comput. Phys.*, 151(2):498–533, 1999.

[Suc01] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.

[Sus03] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics*, 187(1):110–136, 2003.

[SW02] S. Schaefer and J. Warren. Dual contouring: "the secret sauce". Technical report, Rice University, 2002.

[SWB00] Jeffrey Smith, Andrew Witkin, and David Baraff. Fast and controllable simulation of the shattering of brittle objects. In *Graphics Interface*, pages 27–34, May 2000.

[SY04] L. Shi and Y. Yu. Visual smoke simulation with adaptive octree refinement. In *Computer Graphics and Imaging*, 2004.

[SY05] Lin Shi and Yizhou Yu. Taming liquids for rapidly changing targets. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 229–236, New York, NY, USA, 2005. ACM Press.

[TBHF03] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 68–74. Eurographics Association, 2003.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Computer Graphics*, SIGGRAPH 88 Proceedings, pages 269–278. ACM Press, 1988.

[TFK+03] Tsunemi Takahashi, Hiroko Fujii, Atsushi Kunimatsu, Kazuhiro Hiwada, Takahiro Saito, Ken Tanaka, and Heihachi Ueki. Realistic animation of fluid with splash and foam. *Journal of Computer Graphics (Proceedings of Eurographics 2003)*, 22(3):391–400, September 2003.

[THM+03] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranerts, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision, Modeling, Visualization VMV*, pages 47–54, 2003.

[THMG04] Matthias Teschner, Bruno Heidelberger, Matthias Müller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *Proceedings of Computer Graphics International (CGI)*, pages 312–319, Jun 2004.

[Thü06] Nils Thürey. Fluid simulation with blender. *Dr. Dobb's Journal*, January 2006.

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, March 2005.

[TKPR06] Nils Thürey, Richard Keiser, Mark Pauly, and Ulrich Rüde. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, 2006. To appear.

[TKR05] Nils Thürey, Carolin Körner, and Ulrich Rüde. Interactive free surface fluids with the lattice boltzmann method. Technical Report 05–4, Computer Science Department, Friedrich-Alexander-Universitä Erlangen-Nürnberg, 2005.

[TNY85] H. Takewaki, A. Nishiguchi, and T. Yabe. The cubic-interpolated pseudo-particle (cip) method for solving hyperbolic-type equations. *Journal of Computational Physics*, 61, 1985.

[Ton91] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface*, pages 255–262, June 1991.

[Ton92] David Tonnesen. Spatially coupled particle systems. In *SIGGRAPH 92 Course 16 notes: Particle System Modeling, Animation, and Physically Based Techniques*, pages 4.1–4.21, 1992.

[Ton98] David Tonnesen. *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. PhD thesis, University of Toronto, November 1998.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics*, SIGGRAPH 87 Proceedings, pages 205–214, July 1987.

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (from goop to glop). In *Graphics Interface '89*, pages 219–226, 1989.

[TR04] Nils Thürey and Ulrich Rüde. Free surface lattice-boltzmann fluid simulations with and without level sets. In *Proceedings of Vision, Modeling, and Visualization (VMV)*, pages 199–208. IOS Press, 2004.

[TRS06] Nils Thürey, Ulrich Rüde, and Marc Stamminger. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM Press, 2006. To appear.

[TSIF05] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 181–190, New York, NY, USA, 2005. ACM Press.

[TT94] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94*, pages 43–50, New York, NY, USA, 1994. ACM Press.

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.

[TW90] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation*, 1(1):73–80, 1990.

[TY87] Hideaki Takewaki and Takasi Yabe. The cubic-interpolated pseudo particle (cip) method: application to nonlinear and multi-dimensional hyperbolic equations. *Journal of Computational Physics*, 70(2):355–372, 1987.

[van97] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[VB02] J. Villard and H. Borouchaki. Adaptive meshing for cloth animation. In *11th International Meshing Roundtable*, pages 243–252, Ithaca, New York, USA, 15–18 September 2002. Sandia National Laboratories.

[VMT97] Pascal Volino and Nadia Magnenat-Thalmann. Developing simulation techniques for an interactive clothing system. In *Proceedings of the 1997 International Conference on Virtual Systems and MultiMedia*, page 109, Washington, DC, USA, 1997. IEEE Computer Society.

[Wat87] Keith Waters. A muscle model for animation three-dimensional facial expression. In *SIGGRAPH '87*, pages 17–24, New York, NY, USA, 1987. ACM Press.

[WDGT01] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum (Proceedings of Eurographics 2001)*, 20:349–358, September 2001.

[WH94] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM Press, 1994.

[WH04] Mark Wiebe and Ben Houston. The tar monster: Creating a character with fluid simulation. In *Proceedings of the SIGGRAPH 2004 Conference on Sketches & Applications*. ACM Press, 2004.

[WHP+06] Martin Wicke, Philipp Hatt, Mark Pauly, Matthias Müller, and Markus Gross. Versatile virtual materials using implicit connectivity. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, July 2006. to appear.

[WLG04] Stephan Würmlin, Edouard Lamboray, and Markus Gross. 3D video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics*, 28(1):3–14, 2004.

[WLMK04] Xiaoming Wei, Wei Li, Klaus Mueller, and Arie E. Kaufman. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):164–176, 2004.

[WLW+05] Stephan Würmlin, Edouard Lamboray, Michael Waschbüsch, Peter Kaufmann, Aljoscha Smolic, and Markus Gross. Image-space free-viewpoint video. In *Vision, Modeling, Visualization VMV'05*, 2005.

[WMW86] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2, 1986.

[WPK+04] Tim Weyrich, Mark Pauly, Richard Keiser, Simon Heinzle, Sascha Scandella, and Markus Gross. Post-processing of scanned 3d surface data. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 85–94, 2004.

[WSG05] Martin Wicke, Denis Steinemann, and Markus Gross. Efficient animation of point-based thin shells. In *Proceedings of Eurographics '05*, pages 667–676, 2005.

[WT91] K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *Journal of Visualization and Computer Animation*, 2(2):123–128, 1991.

[WTG04] Martin Wicke, Matthias Teschner, and Markus Gross. CSG tree rendering of point-sampled objects. In *Proceedings of Pacific Graphics 2004*, pages 160–168, 2004.

[WZF+03] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, and A. Kaufman. Blowing in the wind. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 75–85. Eurographics Association, 2003.

[WZF+04] Xiaoming Wei, Ye Zhao, Zhe Fan, Wei Li, Feng Qiu, and Suzanne Yoakum-Stover. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, 2004.

[YT02] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. In *IEEE Transactoins on Visualization and Computer Graphics*, pages 346–359, 2002.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005)*, 24(3):965–972, 2005.

[ZPKG02] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: an interactive system for point-based surface editing. *ACM Transactions on Graphics (SIGGRAPH 2002 Proceedings)*, pages 322–329, 2002.

[ZRB+04] Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. Perspective accurate splatting. In *Proceedings of Graphics Interface*, pages 247–254, 2004.

Copyrights

Several figures published in this dissertation reproduce image and data sets courtesy of the following companies and institutions:

- The goblinko model on page ii is copyright of Turbo Squid, Inc.
- The image in Figure 3.2 is courtesy of Matthias Müller, AGEIA/NovodeX.
- The image in Figure 3.3 is courtesy of Robert Bridson, UBC.
- The image in Figure 3.4 is courtesy of Mark Carlson, DNA Productions, Inc.
- The image in Figure 3.5 is courtesy of Tolga G. Goktekin, UC Berkeley.
- The duck model used in Figures 4.3 and 4.7 is copyright of Turbo Squid, Inc.
- The horse model used in Figure 4.8 is copyright of Robert W. Sumner, ETH Zurich.
- The Max Planck model used in Figures 5.1, 5.2, 5.5, 5.21, 5.23 and 6.12 is copyright of the MPI.
- The pharaoh model used in Figures 5.7 and 6.9 is copyright of Turbo Squid, Inc.
- The Igea model used in Figures 5.8, 5.19, 6.7 and 6.12 is copyright of Cyberware, Inc.
- The dragon head model used in Figure 5.22 is copyright of the Stanford Computer Graphics Group.
- The Santa Claus model used in Figure 5.22 is copyright of Cyberware, Inc.
- The octopus model used in Figure 5.23 is copyright of Mark Pauly, ETH Zurich.

Curriculum Vitae

Personal Data

Name	Richard Keiser
E-Mail	keiser@inf.ethz.ch
Address	Computer Graphics Laboratory ETH Zentrum Haldeneggsteig 4 8092 Zurich Switzerland `http://graphics.ethz.ch/~rkeiser/`
Date of Birth	January 20, 1978
Nationality	Swiss

Education

Jun. 2003 – Aug. 2006	Ph.D. Student at the Computer Graphics Laboratory and Applied Geometry Group Computer Science Department, ETH Zurich, Switzerland
Aug. 2005 – Sep. 2005	Visiting Researcher at the Leonidas Guibas Laboratory Computer Science Department, Stanford University, USA
Jul. 2004 – Sep. 2004	Visiting Researcher at the Leonidas Guibas Laboratory Computer Science Department, Stanford University, USA
Apr. 2003	Diploma Degree in Computer Science (ETH Medal) Computer Science Department, ETH Zurich, Switzerland
Oct. 1998 – Mar. 2003	Undergraduate Studies in Computer Science Computer Science Department, ETH Zurich, Switzerland

Scientific Publications

ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006
Nils Thürey, Richard Keiser, Mark Pauly, Ulrich Rüde: *Detail-Preserving Fluid Control*.

Computer Graphics Forum 2006
Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxermann, Mark Carlson: *Physically Based Deformable Models in Computer Graphics*.

ETH CS Technical Report 520, 2006
Richard Keiser, Bart Adams, Leonidas J. Guibas, Philip Dutré, Mark Pauly: *Multiresolution Particle-Based Fluids*.

ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005
Matthias Müller, Barbara Solenthaler, Richard Keiser, Markus Gross: *Particle-Based Fluid-Fluid Interaction*.

Eurographics Symposium on Point-Based Graphics 2005
Richard Keiser, Bart Adams, Dominique Gasser, Paolo Bazzi, Philip Dutré, Markus Gross: *A Unified Lagrangian Approach to Solid-Fluid Animation*.

Eurographics 2005
Bart Adams, Richard Keiser, Mark Pauly, Leonidas J. Guibas, Markus Gross, Philip Dutré: *Efficient Raytracing of Deforming Point-Sampled Surfaces*.

ACM SIGGRAPH 2005
Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, Leonidas J. Guibas: *Meshless Animation of Fracturing Solids*.

Vision, Modeling & Visualization 2004
Richard Keiser, Matthias Müller, Bruno Heidelberger, Matthias Teschner, Markus Gross: *Contact Handling for Deformable Point-Based Objects*.

Vision, Modeling & Visualization 2004
Bruno Heidelberger, Matthias Teschner, Richard Keiser, Matthias Müller, Markus Gross: *Consistent Penetration Depth Estimation for Deformable Collision Response*.

ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004
Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, Marc Alexa: *Point Based Animation of Elastic, Plastic and Melting Objects*.

Eurographics Symposium on Point-based Graphics 2004
Tim Weyrich, Mark Pauly, Richard Keiser, Simon Heinzle, Sascha Scandella, Markus Gross: *Post-Processing of Scanned 3D Surface Data*.

Eurographics 2003

Mark Pauly, Richard Keiser, Markus Gross: *Multi-Scale Feature Extraction on Point-Sampled Surfaces*.

ACM SIGGRAPH 2003

Mark Pauly, Richard Keiser, Leif Kobbelt, Markus Gross: *Shape Modeling with Point-Sampled Geometry*.

Diploma thesis, Computer Graphics Laboratory, ETH Zurich, 2003

Richard Keiser: *Collision Detection and Response for Interactive Editing of Point-Sampled Models*.

Semester thesis, Computer Graphics Laboratory, ETH Zurich, 2002

Richard Keiser: *Feature Detection and Reconstruction on Point-Sampled Models*.