THE SIMULATION AND VISUALIZATION OF MULTI-PHASE FLUID

Fang Chen

Thesis approved by the Department of Computer Science of the
University of Kaiserslautern (TU Kaiserslautern) for the award of the
Doctoral Degree of Engineering (Dr.-Ing)

Date of Ph.D. defense: 22/06/2012

Supervisors:

Prof. Dr. Hans Hagen

Prof. Dr. Bernd Hamann

D 386

Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from November, 2008 to December, 2011, under the supervision of Prof. Hans Hagen at the University of Kaiserslautern.

Fang Chen
Kaiserslautern, Germany
2012

Curriculum Vitae

Fang Chen
IRTG 1131
Department of computer science,
University of Kaiserslautern.
Email: chen@informatik.uni-kl.de

Education:

University of Kaiserslautern, Germany Ph.D. candidate. Simulation and visualization of multi-phase fluid.	2008-2012
University of Eindhoven, the Netherlands Master of Science. Majored in industrial and applied mathematics.	2006-2008
University of Wuhan, China Bachelor of Science. Majored in applied mathematics.	2002-2006

Experiences

Teaching assistant for the lecture algorithm geometry	2011
Teaching assistant for the lecture scientific visualization	2011
Teaching assistant for the lecture scientific visualization	2009/2010

Research Interests

Computational fluid dynamics. Multi-phase fluid simulation. Fluid surface stability. Topology of fluid interfaces.

To my friends and family.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Hans Hagen, who offered me a great opportunity to work at an excellent international research group with a wonderful working atmosphere. Throughout my research, he provided me with endless inspiration, encouragement, sound supervision, as well as the skills to collaborate with people from different fields. I learnt a lot from him.

I would also like to thank Prof. Bernd Hamann, for his helpful guidance via numerous Skype-meetings.

I am also very grateful to my collaborators from the Center for Mathematical and Computational Modelling (CMCM), who made our collaboration smooth and fruitful.

Last but not least, i would like to thank my entire family and my friends, who assisted me in many different ways.

Abstract

The high complexity of fluid extraction process which involves droplet coalescence, breakup, mass transfer, and counter-current flow often makes design difficult. While the simulation output of extraction columns provides a mathematical understanding of how fluids are mixed inside a mixing device, intuitive and physically plausible visualization techniques are in high demand to help chemical engineers to explore and analyze bubble column simulation data. Among multi-fluid researchers, fluid interfaces and free surfaces are topics of growing interest in the field of multi-phase computational fluid dynamics. Texture based visualisation have proven to be a useful tool in analysing complex time dependent fluid interfaces. In addition, High-level visualization techniques such as topology-based methods can reveal the hidden structure underlying simple simulation data. In this thesis, we first presents the joint research on droplet simulation within an extraction column. The major contribution of this work lies in the visualisation of the multi-fluid simulation. We presents the visualization of such fluid simulation from three perspectives, namely the stochastic visualisation of bubble dynamics, the fluid interface stability and the topology of fluid material interfaces.

Contents

CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

Fluid extraction is a typical chemical process where two types of fluids are mixed together. The high complexity of this process which involves droplet coalescence, breakup, mass transfer, and counter-current flow often makes design difficult. The industrial design of these processes is still based on expensive mini-plant and pilot plant experiments. Therefore, there is a strong need for research into the stimulation of fluid-fluid interaction processes using computational fluid dynamics (CFD).

Previous multi-phase fluid simulations have focused on the development of models that couple mass and momentum using the Navier-Stokes equation. Recent population balance models (PBM) have proved to be important methods for analyzing droplet breakage and collisions. A combination of CFD and PBM facilitates the simulation of flow property by solving coupling equations, and the calculation of the droplet size and numbers. In our study, we successfully coupled an Euler-Euler CFD model with the breakup and coalescence models proposed by Luo and Svendsen (59).

The simulation output of extraction columns provides a mathematical understanding of how fluids are mixed inside a mixing device. This mixing process shows that the dispersed phase of a flow generates large blobs and bubbles. Current mathematical simulation results often fail to provide an intuitive representation of how well two different types of fluid interact, so intuitive and physically plausible visualization techniques are in high demand to help chemical engineers to explore and analyze bubble column simulation data. In chapter 3, we present the visualization tools we developed for extraction column data.

Fluid interfaces and free surfaces are topics of growing interest in the field of multiphase computational fluid dynamics. However, the analysis of the flow field relative to the material interface shape and topology is a challenging task. In chapter 5, we present a technique that facilitates the visualization and analysis of complex material interface behaviors over time. To achieve this, we track the surface parameterization of time-varying material interfaces and identify locations where there are interactions between the material interfaces and fluid particles. Splatting and surface visualization techniques produce an intuitive representation of the derived interface stability. Our results demonstrate that the interaction of a flow field with a material interface can be understood using appropriate extraction and visualization techniques, and that our techniques can help the analysis of mixing and material interface consistency.

In addition to texture-based methods for surface analysis, the interface of two-phase fluid can be considered as an implicit function of the density or volume fraction values. High-level visualization techniques such as topology-based methods can reveal the hidden structure underlying simple simulation data, which will enhance and advance our understanding of multi-fluid simulation data. Recent feature-based visualization approaches have explored the possibility of using Reeb graphs to analyze scalar field topologies(19, 107). In chapter 6, we present a novel interpolation scheme for interpolating point-based volume fraction data and we further explore the implicit fluid interface using a topology-based method.

The main contributions of this thesis are as follows.

- Two-phase fluid simulations were converted to three dimensions. PBMs were combined with conservation laws.

- Stochastic approaches to the visualization of droplet distributions.

- An adaptive approach to droplet path-line integration for capturing droplet breakage and collisions.
- A novel technique to facilitate the visualization and analysis of complex material interface instability over time.
- A physically plausible interpolation scheme for randomly initialized point-based data using volume fractions.
- A topology-based fluid interface analyzer using Reeb graphs.

This thesis is organized as follows.

Chapter 2 presents our joint research on droplet simulation in an extraction column. In chapter 3, we describe the visualization tools we developed to represent the output of an extraction column simulation. An in-depth analysis of fluid interface instability is given in chapter 5. In chapter 6, we further explore fluid interface behavior using a topology-based method. Our conclusions and future work are presented in the last chapter.

1. INTRODUCTION

Chapter 2

Multi-phase fluid simulation

[1]The Navier-Stokes equations (11) were derived from Newton's law to describe the motion of fluids. In mathematical terms, they are partial differential equations that describe the changes of fluid quantities in space and time. According to Newton's law, we normally obtain an equation for mass conservation and another for momentum conservation. In the case of multi-phase fluids, however, a complete set of Navier-Stokes equations is used for each phase of the fluid. The fluids are coupled using further equations such as phase exchange or interface interactions. Recent developments in multi-phase fluid simulation typically exploit the concept of marker functions (5, 96). Existing solvers for multi-phase fluids include a particle-based method (64) smooth particle hydrodynamics (43), and a finite point-set method (49).

In this chapter, we describe the mathematical foundations of multi-phase fluid simulation and the mathematical modeling of an extraction column. In Section 2.1, we introduce the mathematical formulation of a two-phase extraction column problem. Section 2.2 illustrates our choice of mesh handling and the numerical boundary conditions for the simulation.

In chapter 3, we describe the results of our joint research with the Center for Mathematical and Computational Modeling (CMCM). This interdisciplinary research col-

---

[1] Part of the work in this chapter was published in (40, 41, 106)

laboration was aimed at developing a complete framework for modeling, simulating, and visualizing multiphase fluids in an extraction column.

2.1 Mathematical foundations of the extraction column

The simulation of extraction columns using computational fluid dynamics (CFD) started with Rieger et al. (85), who used the CFD program Fire for single-phase simulations of a type of extraction column with a rotating disc contactor (RDC). Modes & Bart (67) used the program FIDAP to perform two-phase simulations. The change in droplet size in an RDC column was considered by Vikhansky & Kraft (105) and Drumm & Bart (29) using population balance models (PBMs). Drumm & Bart (29) focused mainly on moment-based PBMs while Vikhansky & Kraft (105) used the Monte Carlo method. Hlawitschka (42) translated the results of (29) to a three-dimensional test case for a Kühni extraction column. Similar to PBMs, a one-group model (OPOSPM) was used that guaranteed a low computational time and a good prediction of the hydrodynamics and flow field (28). Hlawitschka (42) investigated an adaptation of coalescence and breakage kernels using the models proposed by Martinez-Bazán et al. (63) and Prince & Blanch (82).

We present the mathematical formulation of our extraction column simulation in the following sequence:

In section 2.1.1, we introduce the general Navier-Stokes equations for multi-phase fluid simulations. Section 2.1.2 integrates PBMs into the system, which provides further possibilities for the simulation of bubble movements with variable diameters and quantity. The two-phase fluid simulation was carried out in a DN32 extraction column using the model of Andersson (4) for breakage and an adapted model of Coulaloglou & Tavlarides (25) for coalescence, which are described in detail in section 2.1.3.

2.1.1 Conservation laws for two-phase fluids

We assume that each phase of the fluid is continuous and incompressible. Each phase can be represented using the volume fraction α, which is identical to the statistical probability of the droplets being at a specific position. In each computational cell, the sum of the phase fractions in the continuous phase α_c and dispersed phase α_b must be 1, as follows.

$$\alpha_c + \alpha_d = 1. \tag{2.1}$$

The mass transportation of each phase is described using a conservation equation and they are solved separately for each phase. The continuity equation consists of a storage term and a convective term on the left side of the equation and a source term on the right side:

$$\begin{cases} \dfrac{\partial(\alpha_c \rho_c)}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{u}_c) = \rho_c S_c. \\ \dfrac{\partial(\alpha_d \rho_d)}{\partial t} + \nabla \cdot (\alpha_d \rho_d \mathbf{u}_d) = \rho_d S_d. \end{cases} \tag{2.2}$$

The momentum balance of the continuous phase is given as follows.

$$\frac{\partial(\alpha_c \rho_c)\mathbf{u}_c}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{u}_c \mathbf{u}_c) - \tau_l = -\alpha_c \nabla p + \alpha_c \rho_c \mathbf{g} + \mathbf{F}_c + \rho_c \mathbf{u}_c S_c. \tag{2.3}$$

The left side of the equation shows the rate of change of the momentum while the right side shows the effects of pressure and stress-strain. The interaction between the two phases is considered by the resistance force $\mathbf{F}_c$. The source term S is set to 0. In the dispersed phase, the momentum equation is similar to equation (2.3). Analogous to the work of (29), only the drag force is considered to be interphase resistance. The mass force and buoyancy force can be neglected in liquid-liquid systems. The resistance between the continuous phase and the dispersed phase is calculated based on the model of Schiller & Naumann (89). The drag coefficient C_D is defined as

$$F_{c,d} = \frac{3\rho_c \alpha_c \alpha_d C_d |u_d^\Gamma - u_c^\Gamma|(u_d^\Gamma - u_c^\Gamma)}{4 d_d} \tag{2.4}$$

where

$$C_D = \begin{cases} 24(l + 0.15 Re^{0.678})/Re, & Re \leq 1000 \\ 0.44 & Re > 1000. \end{cases} \tag{2.5}$$

Furthermore, the Reynolds number Re is defined (29, 42) as

$$Re = \frac{\rho_c |u_d^\Gamma - u_c^\Gamma| d_d}{\mu_c}.$$

2.1.2 Population balance models

In this section, we present a model that couples mass transportation and PBMs. Variations in the droplet size and number are taken into consideration by introducing breakage and coalescence kernels.

In addition to the four mass and momentum equations (2.2) (2.3), a population balance equation is added to the system to describe the change in the droplet number and size. In general, the formulation of the population balance equation can be written in terms of the number concentration function $n(V,t)$ (9):

$$\frac{\partial}{\partial t}(\rho_d n(V,t)) + \nabla \cdot (u\rho_d n(V,t)) + \nabla \cdot (G\rho_d n(V,t)) = \rho_d S(V,t). \tag{2.6}$$

The number function $n(V,t)$ is dependent on the volume of a droplet V and the current time t.

$S(V,t)$ is the source term that describes the total number of droplets per unit volume. $S(V,t)$ is variable because of the interactions among droplets and the turbulence of the continuous phase. The source term $S(V,t)$ is described in term of the birth and death rates of the droplets

$$S(V,t) = B^C(V,t) - D^C(V,t) + B^B(V,t) - D^B(V,t). \tag{2.7}$$

The first two terms describe the droplet birth rate and death rate due to collision, while the latter two describe those due to droplet breakage. A detailed formulation is given as follows (9).

$$\begin{cases} B^C(V,t) = \frac{1}{2}\int_0^V a(V-V^{'},V^{'})n(V-V^{'},t)n(V^{'},t)\,dV^{'} \\ B^B(V,t) = \int_0^\infty g(V^{'})\beta(V|V^{'})n(V^{'},t)\,dV^{'} \\ D^C(V,t) = \int_0^\infty a(V,V^{'})n(V,t)n(V^{'},t)\,dV^{'} \\ D^B(V,t) = g(V)n(V,t) \end{cases} \tag{2.8}$$

$a(V,V^{'})$ is known as the aggregation kernel, which is a combination of the collision frequency $h(V,V^{'})$ and the probability of a successful collision $\lambda(V,V^{'})$ between two droplets with volume V and $V^{'}$:

$$a(V,V^{'}) = h(V,V^{'}) \cdot \lambda(V,V^{'}). \tag{2.9}$$

$g(V)$ denotes the breakage frequency, i.e., the g-percent of droplets with volume V that break per time unit. More details of the PBM can be found in (60).

2.1.3 Droplet sizes, breakage, and coalescence

The previous section introduced the general formulation of PBMs. In the above models, however, it is not sufficient to calculate the changes in individual droplets. The variations in droplet size have been studied using various models, such as the class method and the Quadrature Method of Moments (QMOM). In our approach, which is a one group model, the One Primary One Secondary Particle Method is used ((28)). This model calculates the mean droplet size in each cell based on the volumetric diameter, which is defined based on the third and zeroth momentum:

$$d_{30} = \sqrt[3]{\frac{m_3}{m_0}} = \sqrt[3]{\frac{6\alpha}{\pi_0 m_0}}. \tag{2.10}$$

The third momentum refers to the total volume of the droplets while the zeroth momentum gives the total number of droplets.

The number of droplets increases due to breakage and decreases due to coalescence. In addition to the equations above, an additional conservation equation has to be added that describes the droplet concentration in the dispersed phase d,

$$\frac{\partial(\alpha_d m_0 \rho_d)}{\partial t} + \nabla \cdot (\alpha_d \rho_d m_0 \mathbf{u_d}) = \alpha_{\mathbf{d}} \rho_{\mathbf{d}} \mathbf{S_d}. \tag{2.11}$$

The source term on the right side describes the generation and reduction of the droplet number. In this study, we used the coalescence model of (25) and the breakage model of (4). The coalescence of two droplets is given by the coalescence efficiency τ and the coalescence rate h, where

$$\begin{aligned} \tau(d_1,d_2) &= \exp(-C_1 \frac{\mu_c \rho_c \varepsilon}{\delta^2} (\frac{d_1 d_2}{d_1+d_2})^4) \\ h(d_1,d_2) &= C_2 \frac{\varepsilon^{1/3}}{1+\alpha} (d_1+d_2)^2 (d_1^{2/3}+d_2^{2/3})^{1/2}. \end{aligned} \tag{2.12}$$

The constants C_1 and C_2 are adjusted for a fluid mixture of butyl acetate/water. The breakup rate g is given by:

$$g(d) = \int_{d_{0/10}}^{10d_0} \omega_s(d_0,\lambda)P(d_0,\lambda)\,d\lambda. \tag{2.13}$$

The interaction frequency ω_s between a droplet of size d_0 and an eddy of size λ is defined as:

$$\omega_s(d_0,\lambda) = \frac{C_3 \pi d_0^3 \varepsilon^{1/3}}{6\lambda^{14/3}}. \tag{2.14}$$

The constant C_3 is taken from the model of (59), where it is given as 0.822. The probability of an eddy breaking up a fluid particle of size d is given by the integral of the normalized energy distribution $\phi(\chi)$:

$$P(d_0,\lambda) = \int_{\chi_{\min}}^{\infty} \phi(\chi)\,d\chi. \tag{2.15}$$

χ is defined as the ratio of the eddy viscosity relative to the average eddy viscosity.

The breakage and coalescence kernels can be modeled further if fluctuations or surfactants are present. The corresponding parameters for the coalescence and breakage kernels need to be adjusted in these cases.

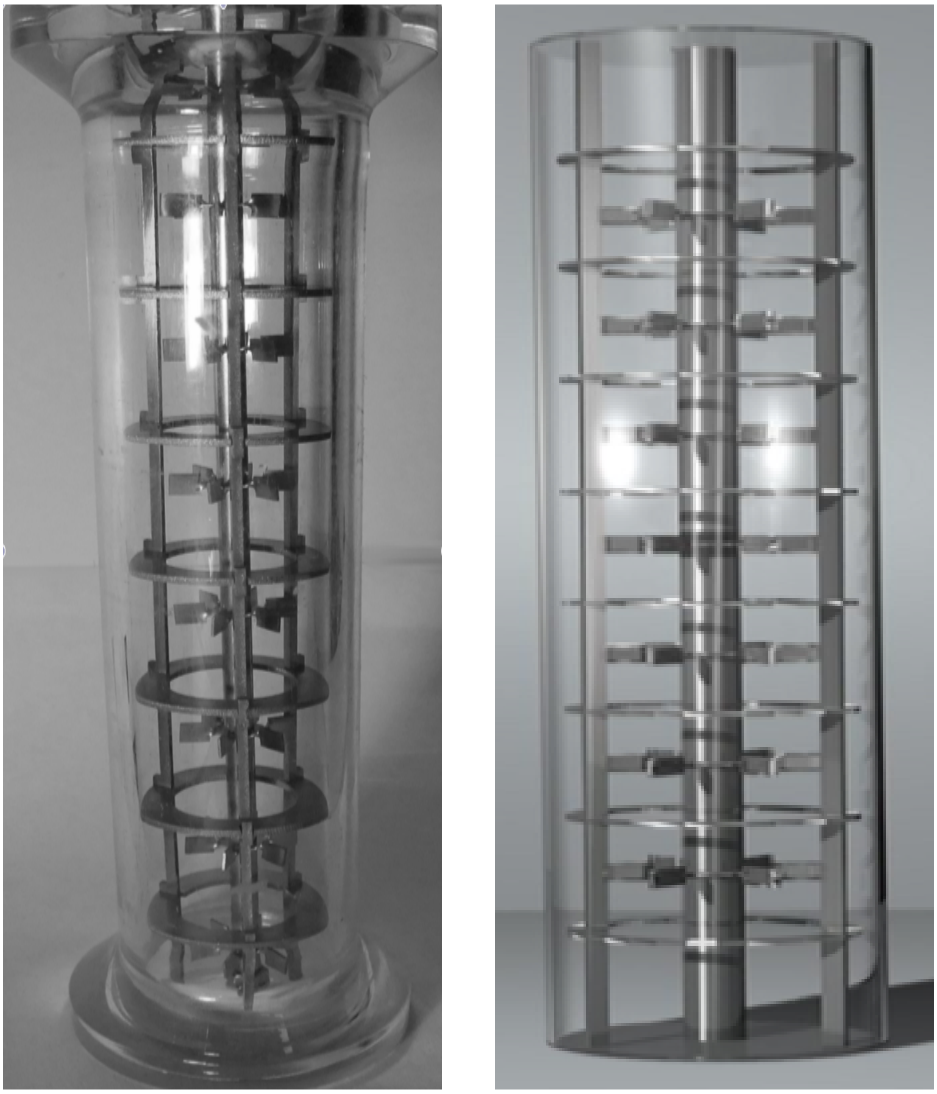
2.2 Geometry and boundary setup for 3D simulation

The numerical mesh setup and boundary conditions are discussed in this section.

A seven-compartment section of a Kühni Miniplant column mesh was rebuilt in Gambit with over 500,000 cells. This mesh was a scaled-down duplicate of the laboratory device. Figure 2.1 shows a comparison between the laboratory device (left) and a realistic rendering of our simulation mesh.

The inflow and outflow are defined at the column bottom and top, respectively. A close-up of the mesh is shown in Fig. 2.2. We treat the mesh parts (pink) around the stirrers as a moving reference frame (MRF). A six-baffled stirrer was installed in the middle of each compartment (Fig. 2.2). The neighboring compartments were separated by two stators, which in this case were two metal rings. Three stream-breakers were orientated at an angle of 120 degrees to each other.

The volume stream of each phase (water and butyl acetate) was set to 8 l/h and the stirring speed was set to 300 rpm. The simulation was performed in FLUENT for 20,000 time steps using a time step size of 0.05 s, and the standard relaxation factors in the commercial CFD code of FLUENT were used. A converged and steady state solution was reached at the end of the simulation.



(a) (b)

Figure 2.1: Left: the extraction column used in our laboratory. Right: the computational mesh rendered using Blender.

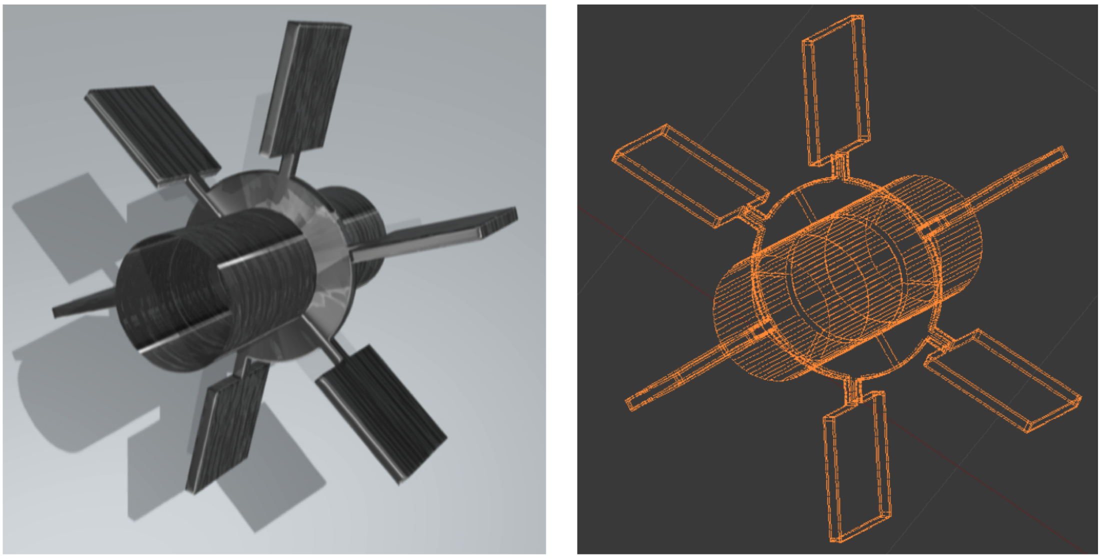


Figure 2.2: Stirrer

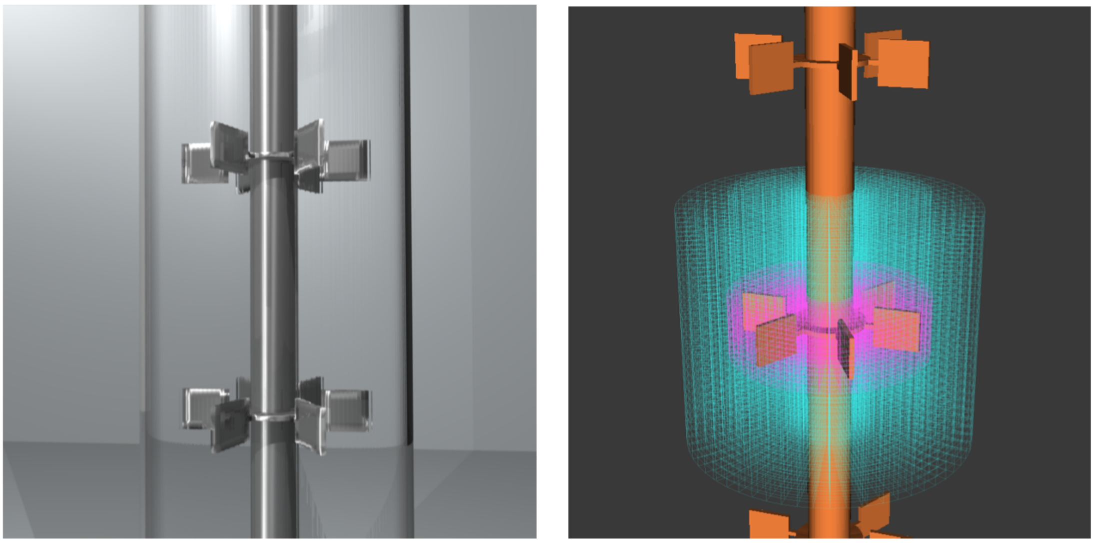


Figure 2.3: Left: the extraction column rendered using Blender. Right: the computational mesh in one compartment.

Chapter 3

Visualization techniques for the extraction column dataset

[1] Domain experts commonly acquire a better understanding of simulation results by visual inspections, which require visualization techniques. The simulation output provides numerical and quantitative results, whereas visualization is a postprocessing step that gives users robust and sophisticated visual insights into the simulated data. Using the visualization methods proposed in this section, domain experts can analyze and study the complex patterns in large-scale time-variable simulation datasets.

The time-variable distribution of droplets needs to be displayed intuitively because it is a key feature of extraction column simulations. One challenge when visualizing two-phase flow is the representation of the dispersed phase, which requires a scheme that couples the cell volume fraction and the droplet size. A recent stochastic modeling study (37) suggested how these two fields can be combined and brought together. In section 3.1, we present a stochastic approach that addresses the problem of coupling the volume fraction and particle diameter.

---

[1] Visualization results this chapter were published in (10, 22, 41)

In addition to single-framed droplet distributions, the dynamic behaviors of droplet sets are of great interest to engineers and researchers. Recent fluid visualization research showed that line integrals can be useful visual tools for tracking fluid particles (34, 56, 86). As well as the accurate computation of line integrals, the start and end points of a droplet path should be handled with special care in the extraction column, because fluid droplets break or merge along their paths. We propose a novel re-seeding approach for capturing droplet path lines in time-variable multi-flow fields. Further exploration of the simulation data is conducted using feature-based visualization techniques. Using our technique, the path lines of fluid droplets can be re-initialized after interrupting the stirring device. Furthermore, droplet breakage is captured using bifurcating path lines.

In this chapter, we present our 3D extraction column data visualizations as follows. In section 3.1, we present a stochastic approach that addresses the bubble distribution problem. The challenge in visualizing the simulation output is that the droplet size and droplet number needs to be coupled. Thus, we use a stochastic approach to address the underlying problem. In section 3.2, we present an adaptive path line integration algorithm that can tracks events such as collisions and breakups. In the last section (3.3), we provide an overview of the visualization tool we developed for 2D finite point-set datasets.

3.1 Stochastic approach to droplet distribution

Postprocessing of the 3D simulation data is aimed at developing visualization techniques that depict the droplet distribution and the dynamics of droplet movements.

As previously mentioned, we are faced with a dilemma when we display droplets as data points. Each single mesh point carries the particle diameter $d30$ and the cell phase volume fraction α. Mathematical simulation provides an estimate of the number of droplets but it does not show the exact locations of these droplets. Thus, we use a

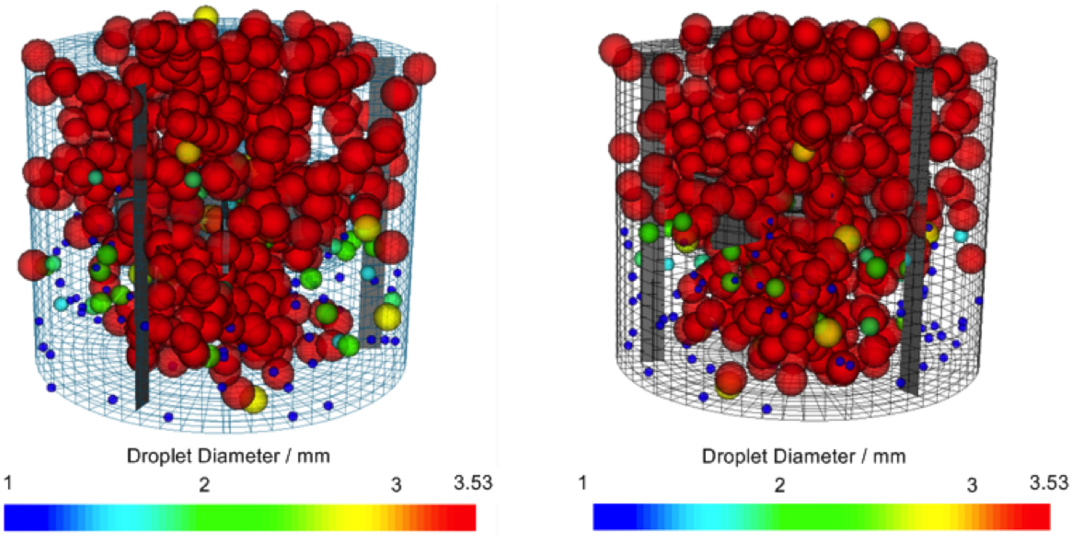


Figure 3.1: Droplet distribution

stochastic distribution to locate the droplets in space and time, as follows.

$$f(x) = \begin{cases} \dfrac{1}{x_{\max} - x_{\min}}, & \forall x \in \text{Volume} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

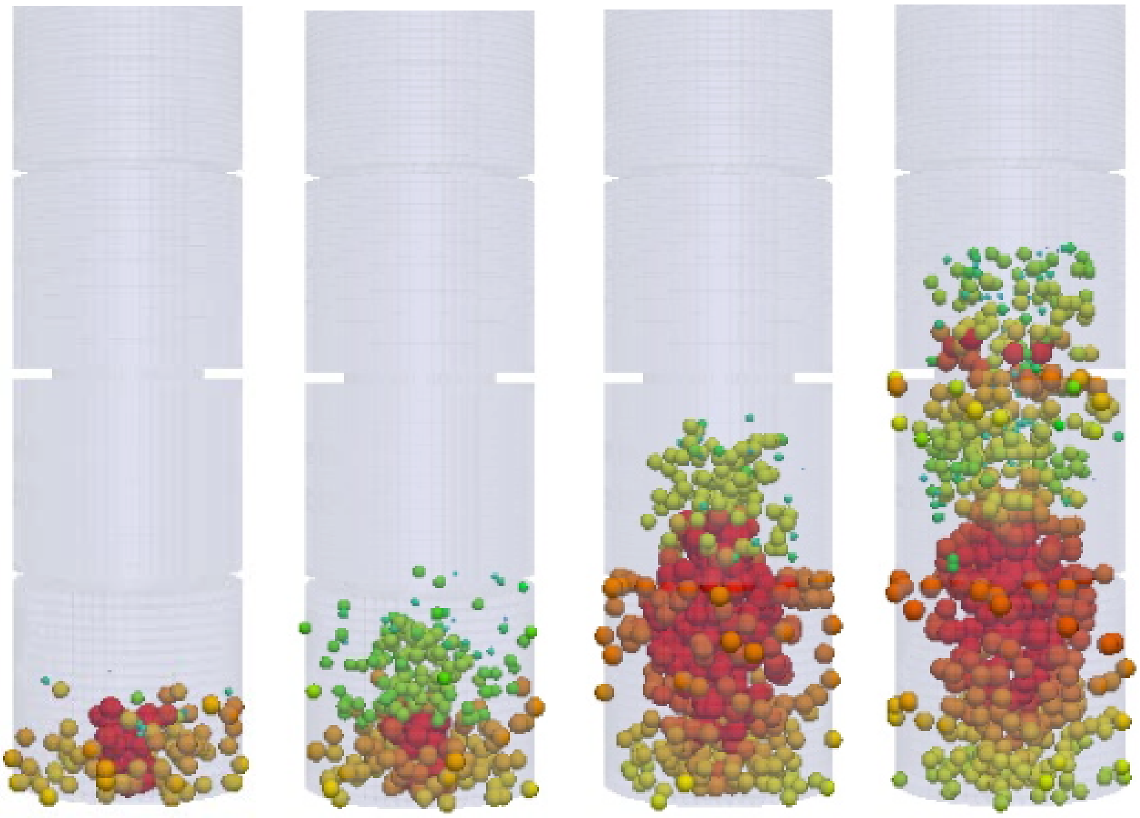
Furthermore, the number of particles in space is estimated using the following equation, so the volume fraction is fulfilled.

$$\sum N \times V_{eachDroplet} = \alpha \times V_{total} \tag{3.2}$$

An example of a droplet distribution visualization in a single column is shown in Figure 3.1. The average droplet diameter is between 1 mm and 3.6 mm. The total droplet distribution is shown in figure 3.2. It can be seen that droplet breakage occurs due to the effect of stirring.

3.2 Adaptive path line integration

In the previous section, we considered the droplet distribution during a single time frame. Thus, we propose an adaptive path line integration algorithm to allow temporal



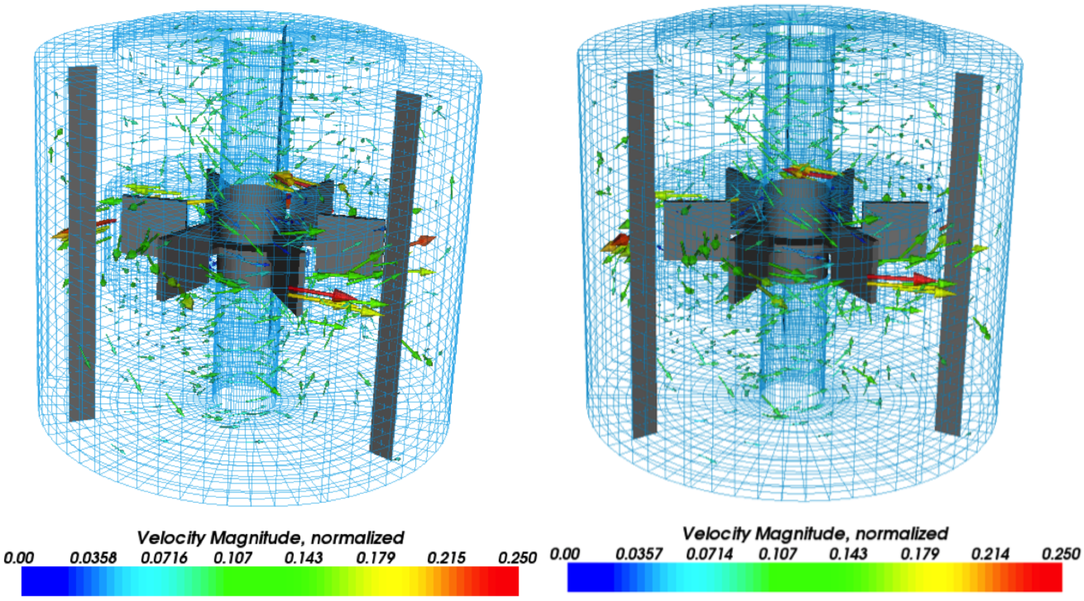
(a) (b) (c) (d)

Figure 3.2: Time sequence of droplets inside the extraction column

visualization, which serves as a tracking tool for the droplet dynamics. Line integrals such as path lines, stream lines, and streak lines are well-known representations used in visualization ((34, 56)). Unlike the single frame visualization of velocity fields (Figure 3.3), path line visualization allows domain experts to track the trajectory of a single droplet, thereby facilitating the temporal analysis of droplet dynamics.

A path line denotes the trace/path of a droplet, which is defined mathematically as:

$$\begin{cases} \frac{\partial}{\partial t} l(x,y,z) = \mathbf{u}(\mathbf{l},\mathbf{t}), \\ l(t_0) = l_0. \end{cases} \tag{3.3}$$



(a) **(b)**

Figure 3.3: Velocity field distribution

However, conventional path line computation has the following disadvantages.

- The path line is interrupted at the mesh boundaries around stirrers where it terminates, and automatic re-seeding is required.
- The path line computation continues even if a droplet vanishes after a certain time period.
- Droplet breakup cannot be captured.

Thus, we propose the following coalescence and breakage detection and re-seeding methods to resolve these problems.

- Re-seeding near stirrers: if a path-line is interrupted at a mesh point near the stirrer, select another point near the stirrer and start a new path line integral from that point.
- Path line termination after droplet interaction: the physical trace of a droplet vanishes when the droplet disappears. Therefore, further integration should be stopped when a large droplet breaks. We incorporate this feature detection method into path line integration, which records the lifespan of a droplet in its path line.
- Droplet collision is detected when two or more path lines intersect.

We now describe the path line termination process in detail. Suppose a droplet has position P_0 at point t_0 after time t_1, and the droplet moves over to P_1, as shown in Figure 3.4. The volume fraction and particle size at each time step are given as α_i , d_i. Using the following detection method, we can test whether t_1 is the time period when a larger droplet breaks.

$$\begin{cases} d_{i+1}/d_i > \sqrt[3]{2} \\ \alpha_i > \alpha_{threshold} \end{cases} \tag{3.4}$$

Condition $d_i + 1/d_i > \sqrt[3]{2}$ guarantees that the volume breaks into two or more equal parts. The threshold check on the volume fraction ensures that a droplet actually exists in the region. The pseudocode of the proposed algorithm is given in Algorithm 1.

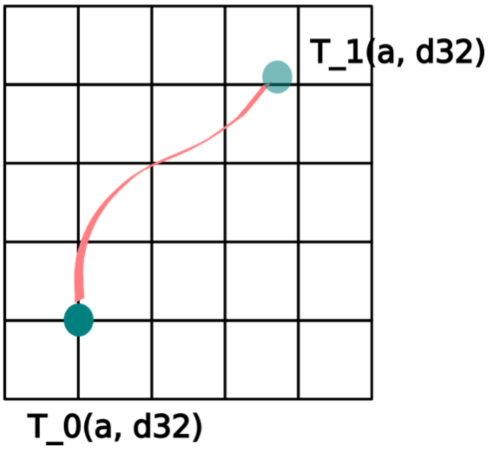


Figure 3.4: Pathline integration

Algorithm 1 Adaptive path line integration with breakup detection

```
for each pathline do
    for $t = 0 \rightarrow t_i$ do
        if $d_{i+1}/d_i > \sqrt[3]{2}$ and $\alpha_i > \alpha_{threshold}$ then
            terminate current path line integration,
            select a neighboring point and start a new path line
        end if
    end for
end for
```

Figure 3.5a shows straightforward integration without feature detection or line re-seeding. This shows that all particle paths are interrupted after they reach the stirrers. Furthermore, the overextension of all the path lines demonstrates that no droplet breakages are recorded.

We implement the proposed re-seeding and termination criteria using the Runge-Kutta 4 integration method in C++. The integration length is set smaller than the average cell size to obtain a smooth line output. Figure 3.5b shows an example of the proposed algorithm. In this case, the path lines are re-seeded after droplet breakage and integration can be continued after interruption of the stirrer.

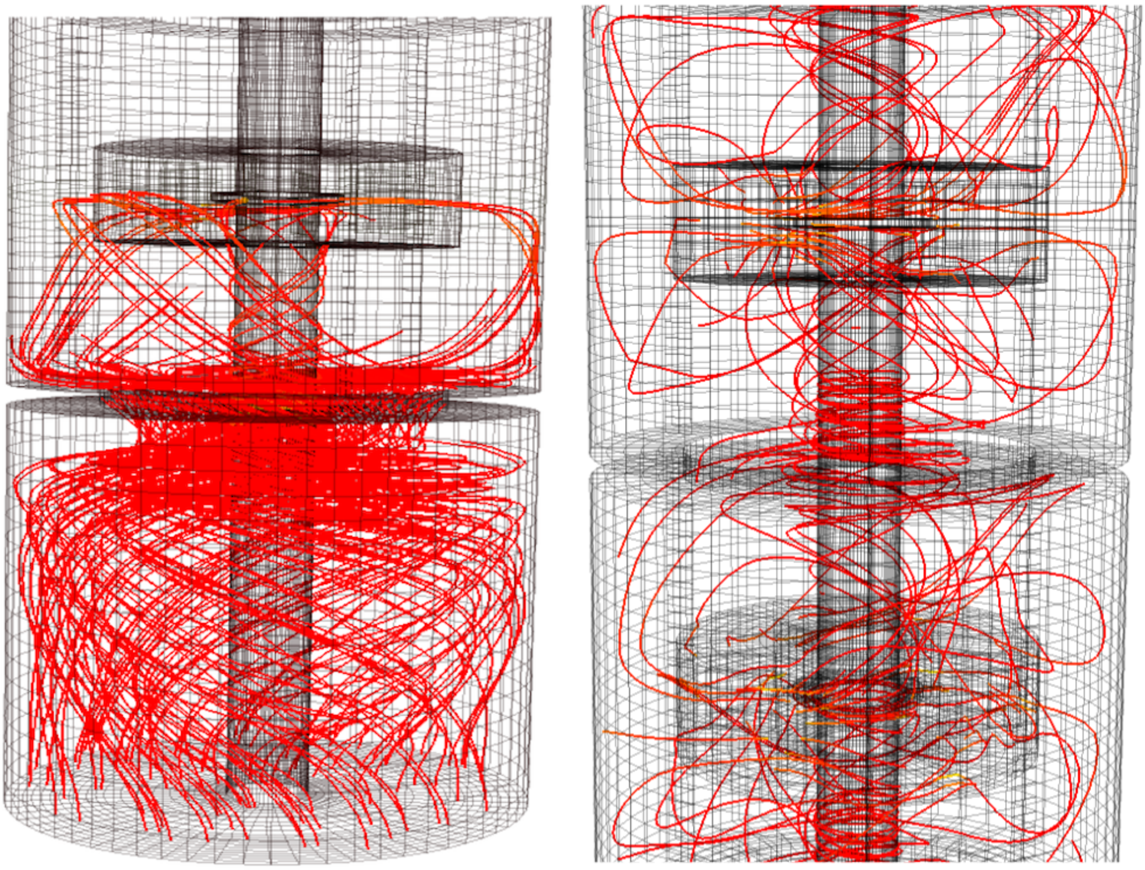
In chapter 2 and chapter 3, we presented an actual example of the workflow when simulating and visualizing two-phase fluid data in a Kühni Miniplant. The main contributions of our research for the visualization community are the development of feature detection and re-seeding algorithms for droplet tracing. Our visualization techniques allow engineers to analyze large-scale simulation outputs effectively. The intuitive representation of the bubble distribution and lifespan provides parameter feedback during extraction column design.

3.3 Visualization of the 2D FPM dataset

In addition to the 3D simulation visualization pipeline, we also developed visualization tools for 2D simulation outputs based on a finite-point set (49). This dataset was provided by Dr Tiwari & Dr Kuhnert at ITWM, Fraunhofer Institute, Kaiserslautern, Germany.

In this section, we briefly describe the visualization tool we designed for visualizing the 2D FPM dataset. We implemented the software using Qt and C++.

The underlying visualization tool allows the user to interactively select part of the mesh and view the mesh using points, lines, or surfaces (see Figure 3.6). We also provide the user with the option of studying the velocity field (figure 3.7) and conducting path line integration (Figure 3.8).



(a) Straightforward path line integration

(b) Path-lines with collision and bifurcation detection

Figure 3.5: Comparison of the straightforward integration method and the proposed method

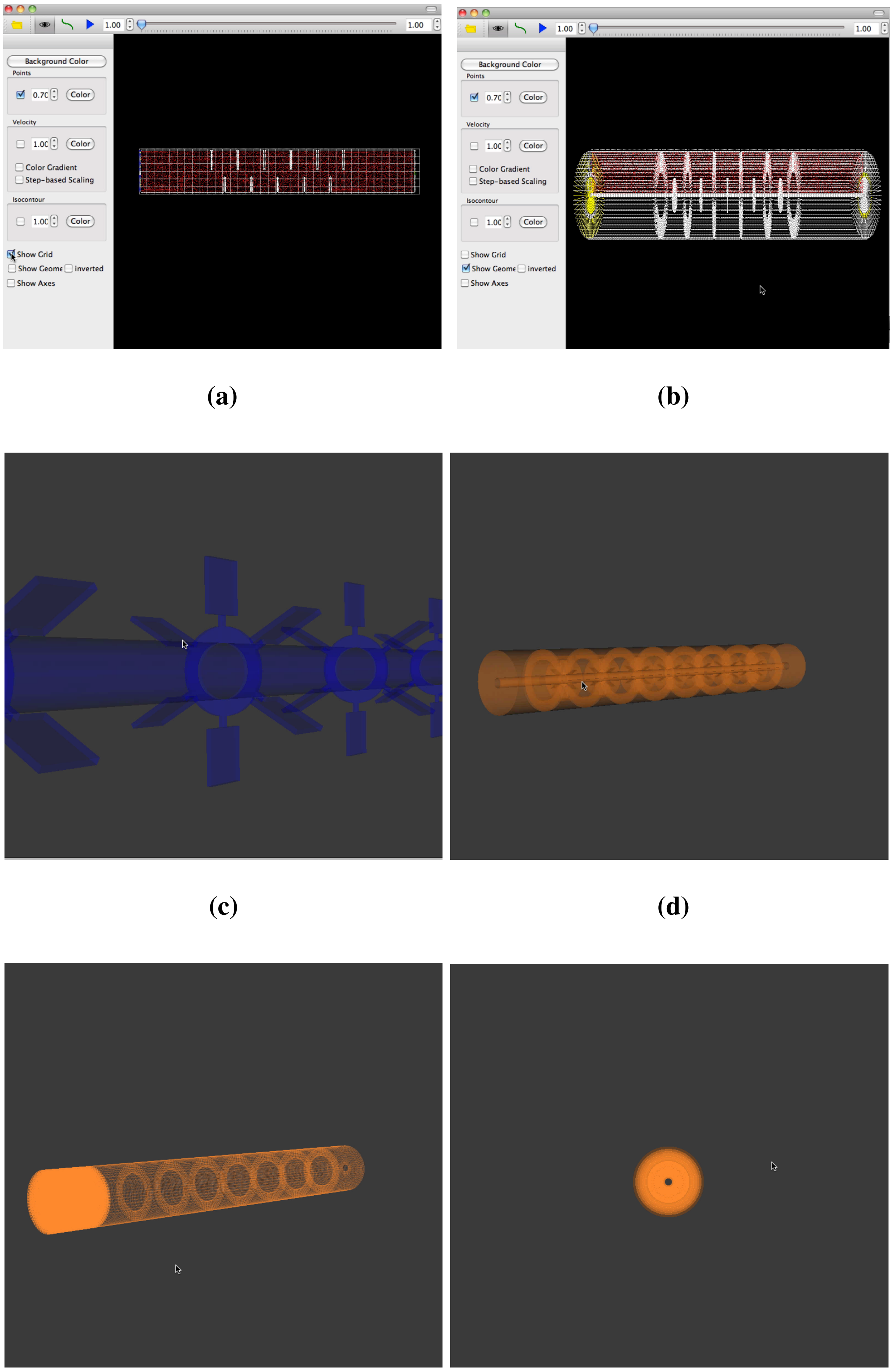
Only the volume fraction field is given for the 2D data-set. In this case, we do not need to apply any special treatments to match the particle diameter. The nature of FPM method means that the point-based mesh does not provide cell information, so the cell volumes are not defined. Therefore, we apply a Voronoi tessellation (6) to the point-cloud, as shown in Figure 3.9b [1]. The cell volume is defined as the area size of the voronoi cell $V_{voronoi}$. Thus, we define the bubble volume as

$$V_{bubble} = \alpha V_{voronoi}.$$

An example of the droplet distribution is shown in Figure 3.10. Furthermore, a Runge-Kutta 4 (2) path line visualization is shown in Figure 3.8.

---

[1] This image was created using the online fortune code at http://www.diku.dk/hjemmesider/studerende/duff/Fortune

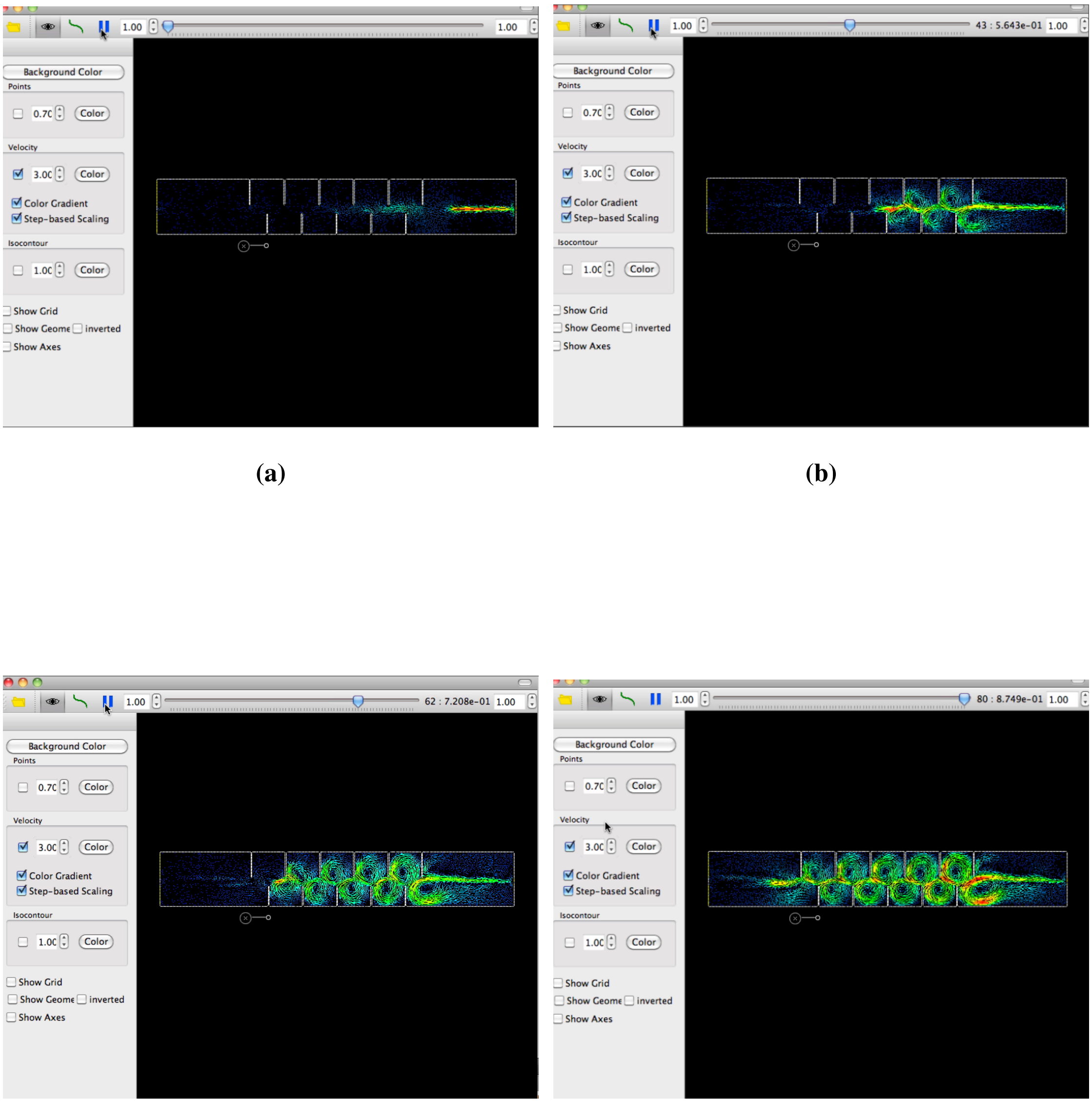


(a) **(b)**

(c) **(d)**

(e) **(f)**

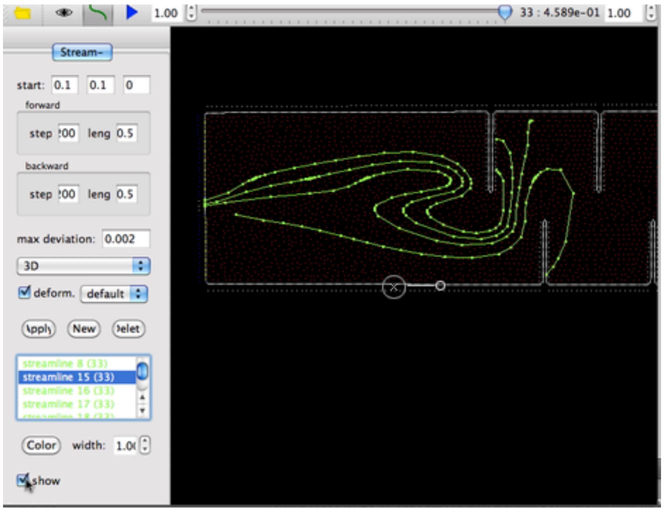
Figure 3.6: A visualizer for 2D FPM data sets.



(a) (b)

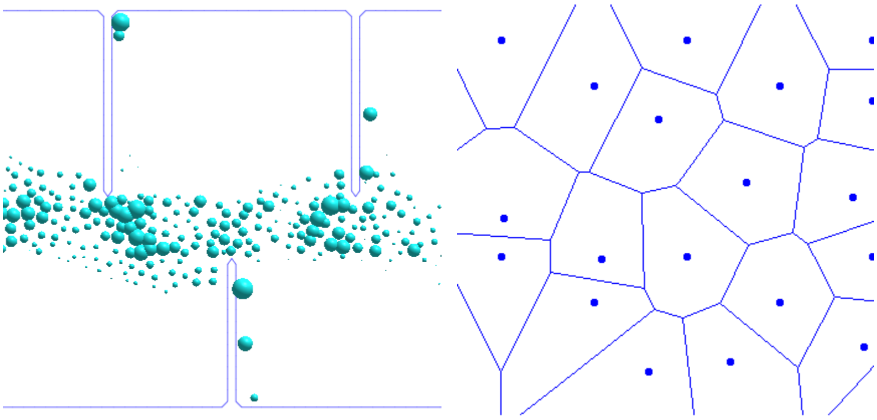
(c) (d)

Figure 3.7: Velocity visualization of the FPM dataset in different time periods.



1.00
33 : 4.589e-01 1.00
start: 0.1 0.1 0
forward
step 200 leng 0.5
backward
step 200 leng 0.5
max deviation: 0.002
3D
☑ deform. default
streamline 8 (33)
streamline 15 (33)
streamline 16 (33)
streamline 17 (33)
width: 1.00
☑ show

Figure 3.8: Stream line integration of a 2D FPM dataset



(a)

(b)

Figure 3.9: a) Close-up of the bubble distribution. b) An example of Voronoi tessellation.

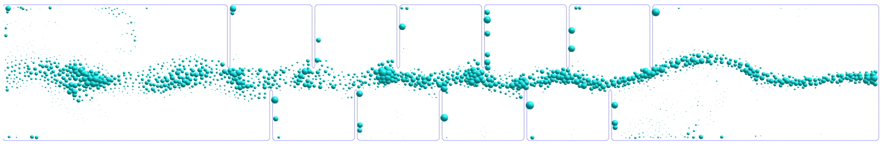


Figure 3.10: Bubble distribution using Voronoi cells.

Chapter 4

Reconstruction and tracking of fluid interfaces

[1]The major features of interest for scientists during multi-fluid simulations are the dynamics of fluid interfaces, known as material boundaries. In the visualization community, one method used to study fluid interfaces is to treat them as continuous integral surfaces. Another perspective when considering fluid interfaces is to study the underlying topological changes in the reconstructed material boundary.

In this chapter, we consider the state of the art in material interface reconstruction and tracking analysis. A survey of existing interface tracking algorithms, including their backgrounds, terms, procedures, and comparisons is given in section 4.1. In section 4.2, we explain the mathematical modeling of multi-phase fluid simulations using a one phase formulation.

---

[1] This chapter is based on our published paper (21)

4.1 State of the art

This section gives an overview of current interface tracking methods. A list of existing techniques for fluid interface reconstruction is given in section 4.1.1, which includes comparisons of their advantages and disadvantages. The level set method (LSM) and volume of fluid (VOF) method are the two main concepts in multi-fluid visualization, and they are introduced in section 4.1.2.

4.1.1 Overview of existing methods

Simulation experts are focused on the mass and momentum transportation of the flow, whereas scientists in the visualization community are interested mainly in locating the fluid interfaces between the components of fluids. Implicit surfaces, dynamic boundaries, and material interfaces are addressed in many publications for marking the interface between two fluids.

Recent studies of interface tracking have focused on the following three methods: the Front Tracking Method (FTM) (102), the LSM (74), and the VOF method (39).

*T*he Front Tracking Method (38, 97, 102) determines the marked interface based on the initial configuration and it assumes that the topology of the interface remains unchanged during the simulation. Unverdi and Tryggvason (102) discretized the flow field by conservative finite difference approximation on a stationary grid and they represented the fluid interface as a separate, unstructured grid that moves through the stationary grid.

In (97), the ghost fluid method was combined with FTM. The proposed method defined fluid interface conditions using the ghost fluid method. An advantage of this method is that nonphysical oscillations are avoided. However, FTM does not automatically keep track of the interface topology. Glimms (35) tried to re-compute the fluid interface while advecting the front grids. This method is limited to topological changes in multiphase fluids, such as the merging or breaking of droplets. To avoid this problem, we propose a new technique for capturing topological changes in fluid interfaces, which we discuss in chapter 6.

*T*he Level Set Method was introduced by Osher (75) in 1988. The material boundary or interface is defined as the zero set of the isocountour or isosurface of the given scalar field (74, 76, 90). Topological changes in the fluid interface can be handled in a natural manner.

Sethian (91) and Lakehal (50) applied LSM to fluid simulation and generalized the idea of a one-phase fluid. In 2002, Enright and Fedkiw et. al. (32) combined Lagrangian marker particles with LSM to provide a smooth geometrical description of the fluid interface. Although LSM is more flexible at handling topology changes in the interface, Müller (69) and Garimella et. al. (33) noted that the material volume is not well-preserved by LSM, which is the main drawback of LSM.

Volume of Fluid Methods (39) is a well-established interface tracking method that is in current use (61, 96). As the name suggests, it keeps the track of the volume of each fluid phase using a computational cell. Therefore, this method is based on subcells or subvolumes. Each phase of the fluid is represented as a volume fraction that indicates the proportion of the cell occupied by this phase. Newton's law of mass conservation is expressed in the form of a volume fraction.

In the VOF method, the fluid interface is implicitly given in cells that occupy fifty percent of each fluid. Further reconstruction is needed to compute the explicit location of the material boundaries.

The reconstruction of the material interface can be achieved by postprocessing the given multi-volume data and it has become an important issue when defining the location of fluid interfaces. Several discrete approaches have been developed by the visualization community for reproducing snapshot-like interfaces. Reconstruction methods are focused mainly on building continuous interfaces from discrete pieces or piecewise functions. By contrast, interface tracking algorithms focus on tracking the dynamic behavior of the interface.

The Simple Line Interface (SLI) method was introduced by Noh and Woodward (71), in which cells are partitioned using simple axis aligned lines or planes to match and preserve their volume. Similar to the SLI method, Youngs (108) and Rider et al. (84) developed a Piecewise Linear Interface Calculation (PLIC) algorithm.

Recent material interface reconstruction approaches have focused on finding a smooth and continuous interface based on fractional material data. Discrete approaches to interface reconstruction were studied by Bonnell et al. (14) and Anderson et al. (3). The first approach constructs boundaries by calculating the intersections between tetrahedrons in material space using Voronoi cells to represent the dominance of one material, whereas the second approach uses a volume-adaptive active interface model to generate high quality boundary meshes.

4.1.2 LSM and VOF

LSM and VOF are used widely for simulating multi-phase fluids or tracking fluid interfaces. During our research, the concepts of marker functions and one phase fluid formulations formed the basis of our new techniques. In the following section, we briefly review the main concepts of LSM and VOF methods.

4.1.2.1 One-phase formulation of a two-phase problem

In the previous chapter, we introduced the two-phase fluid formulation (2.2). However, a one phase fluid model is used widely by the visualization community for decoupling the system and reducing the number of variables.

A detailed description of one-phase fluid formulation can be found in Lakehal (50) and Osher (74). Rather than searching for coupled equation systems, we can reduce the computation complexity by introducing the marker/indicator function χ.

$$\chi = \begin{cases} 1 & \text{in fluid 1} \\ 0 & \text{in fluid 2} \end{cases} \tag{4.1}$$

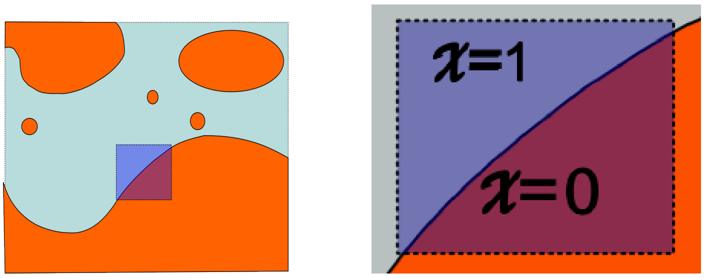
As the name suggests, a marker function indicates a phase change by attaching a marker to one of the phases. In some cases, markers can also be assigned to the exact interface between two materials, as in LSM. Details of LSM are found in section 4.1.2.2.

Rather than having two sets of variables for both phases of the fluid mixture, a marker function can mark the integration volume (if we consider a small volume element, as shown in Fig. 4.1b) based on a characteristic function. Thus, we obtain only one set of conservation laws for the two phase (50) problem:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{4.2}$$

where $\rho = \rho_1 + (\rho_1 - \rho_2)\chi$, and $\mathbf{u}$ is the interfacial velocity given by

$$\mathbf{u} = \rho_1 \mathbf{u}_1 + \rho_2 \mathbf{u}_2.$$



(a) Two phase.

(b) One fluid formulation. Close-up of the volume element

Figure 4.1: One and two phase formulation

The choice of marker function is not always the same. For applications, the marker function is defined by (73) in the following form:

$$\chi = \begin{cases} 1 & \text{in fluid 1} \\ -1 & \text{in fluid -1} \end{cases} \tag{4.3}$$

and in (100):

$$\chi = \begin{cases} 2 & \text{in fluid 1} \\ 1 & \text{in fluid -1} \end{cases} \tag{4.4}$$

Furthermore, the surface normal $\vec{n}$ and curvature κ can be computed with the smoothed marker function $\tilde{\chi}$ using the continuum surface tension model (15, 100):

$$\tilde{\chi} = \frac{\sum \omega_i \chi_i}{\sum \omega_i}. \tag{4.5}$$

The unit normal of the surface was represented by (99) as follows.

$$\vec{n} = \frac{\nabla \tilde{\chi}}{|\nabla \tilde{\chi}|} \tag{4.6}$$

A detailed description of surface tension models can be found in (15, 95).

4.1.2.2 Level set method

In 1988, the American mathematicians Osher and Sethian developed LSM (75) as a simple and versatile method for interface computation and analysis. A fast PDE-based LSM approach was introduced by Peng (80) that require no allocation of exact interfaces. Duan (31) proposed an improved shape optimization for LSM that was capable of handling large time steps. The concept of LSM has since become a very popular in the field of CFD and image processing. An overview of LSM and recent developments was provided by Osher and Fredkiw (76).

In this section, we introduce the general concept of LSM, which is fundamental to our visualization approaches.

Like the VOF method, LSM exploits the concept of one fluid simulation. The implicit material boundary/interface is given by the zero set Γ of the scalar field ϕ:

$$\Gamma : \{(x,y,z)|\phi(x,y,z) = 0\}$$

The scalar function is set to zero at the interface. The scalar function has different signs inside and outside the boundary:

$$\phi = \begin{cases} > 0 & \text{in fluid 1} \\ < 0 & \text{in fluid 2} \\ = 0 & \text{at boundary } \Gamma \end{cases} \tag{4.7}$$

The marker function ϕ in equation (4.7) is similar to that in equation (4.1). In LSM, $\phi = 0$ points to the exact location of the interface. The norm $|\phi(x)|$ generally corresponds to the distance between a point x and the interface. The evolution of the zero sets satisfies the topology equation (4.8). Finding the initial zero sets normally involves extracting the isocontour or isosurface at the start time.

$$\phi_t + \vec{v}|\nabla\phi| = 0. \tag{4.8}$$

For a dataset containing the volume fraction information α, we consider α to be the scalar function ϕ while the fluid interface is defined as cells where $\alpha = 0.5$.

The normal of the interface is given by the equation 4.6 while the curvature of the level set was given in (32, 50, 74), as follows.

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} \tag{4.9}$$

Constraints on the surface curvature should be considered when performing physically correct simulations. A general concern with curvature is the minimization of the surface energy, which is also related to the surface tension. LSM is the most widely used method for identifying isosurfaces or fluid interfaces. It has a simple mathematical formulation and it is easy to solve. It is popular in many disciplines such as medical visualization, image processing, and multi-fluid visualization. However, LSM has the disadvantage that the volume is not always preserved while advecting the interface. This disadvantage can be overcome by applying a volume correction after each numerical advection.

Most current approaches solve the level set equation (4.8) using grid-based numerical methods. A semi- Lagrangian method was proposed by Enright et al. (32) that achieved a fast first order accuracy and improved the mass conservation.

4.1.2.3 Volume of fluid method

In contrast to the nonvolume-preserving LSM, Hirt and Nichols (39) introduced the VOF method in 1988, which ensures the conservation of mass. Sussman and Puckett (96) presented a hybrid LSM/VOF method, which had the goal of higher accuracy for flows where the surface tension and surface topology were the dominant features. Linear approximation is widely used during the reconstruction of VOF interfaces, as in (62). In addition to the conventional first order numerical schemes, Pilliod (46) proposed design criteria to ensure second order accuracy with VOF.

VOF algorithms also employ one phase fluid formulation. A fraction variable C is defined as the integral of the marker function χ in the controlled volume V:

$$C = \frac{1}{V} \int_V \chi(x,t) dV \tag{4.10}$$

In general, the controlled volume V is the volume of a computational cell.

A zero value of C indicates that the cell is fully occupied by fluid type one. If $C = 1$, the cell is filled fully by the other fluid. If $0 < C < 1$, the line/surface separating the two fluids is across this cell.

Given a velocity field V, the volume fraction function should satisfy the transportation theory, as follows.

$$\frac{dC}{dt} + \vec{v} \cdot \nabla C = 0 \tag{4.11}$$

VOF does not simply compute the advection of the fluid because it also involves the approximation and reconstruction of the material interface at certain time periods. This method is accurate in terms of volume preservation, but the interface reconstruction process makes it difficult to maintain the topology of the interface.

Pillido (46) summarized the four main reasons why VOF algorithms are effective: the natural conservation of mass, coincidence of the interface and density jump, the handling of global topology change, and local updates of the interface location.

There are a number of numerical methods for solving the governing equations (4.11). Some are grid-based methods where the computational domain is represented using a structured mesh grid, whereas others use particles to characterize the flow property.

In the following section, we introduce several numerical approaches for solving the governing equations (4.8) and (4.11).

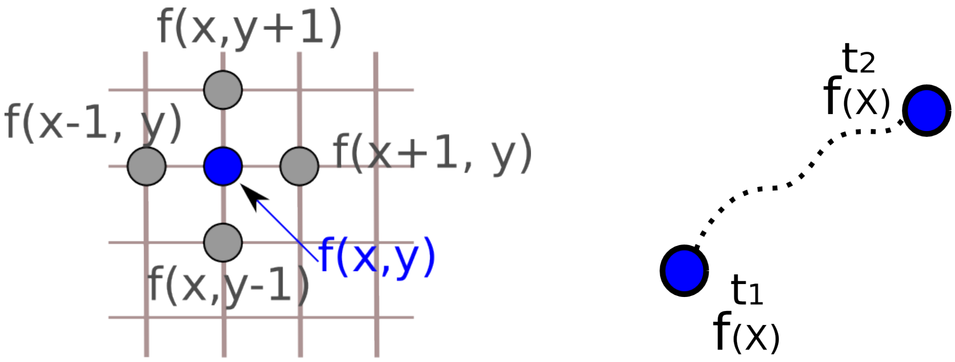
4.2 Numerical methods for solving the governing equations

Depending on the frame of reference, the flow properties can be considered in two different ways for computational fluid dynamics, i.e., the fixed frame–Eulerian specification and the co-moving frame–Lagrangian specification.

- Eulerian specification: in this fixed reference frame (see fig. 4.2a), the grid points remain unchanged over time. The flow property is generally recalculated based on neighboring points. This type of method is normally computationally expensive. To advect the flow quantity, we have to update each grid point based on the previous quantity at this point and its four neighboring points.

- Lagrangian specification: the Lagrangian specification of flow dynamics is also known as particle-based specification, see fig. 4.2b. An observer moves with the particle in the flow direction and measures changes in the flow quantities. Changes in the flow properties are related to the *material derivative*:

$$\frac{\mathrm{D}f}{\mathrm{D}t} = \frac{\partial f}{\partial t} + \vec{v} \cdot \nabla f$$



(a) Fixed mesh grid, Euler types of methods

(b) Co-moving points, Lagrangian type

Figure 4.2: Eulerian and Lagrangian methods

With the Lagrangian method, no connectivity information is provided or needed from the neighboring mes -points.

4.2.1 Grid-based method

The classical way of handling hyperbolic partial differential equations is to solve them numerically using a fixed grid of points. These types of methods solve partial differential equations in a Eulerian manner. To solve the governing parabolic equations for VOF and LS methods, we can apply standard advection schemes such as the upwind scheme (26):

$$\frac{\phi_i^{t+1}-\phi_i^t}{\Delta t}+\vec{v}\,\frac{\phi_i^{t+1}-\phi_{i-1}^t}{\Delta x}=0,$$

and the central difference scheme (55):

$$\frac{\partial\phi}{\partial x}|_i=\frac{\phi_{i+\frac{1}{2}}-\phi_{i-\frac{1}{2}}}{h},$$

where h is a full step length. To avoid dissipation, a second order scheme with flux limiters may also be considered. A large body of literature discusses the computational accuracy and convergence such as (36, 39, 74, 76, 76).

4.2.2 Particle-based method

To solve the hyperbolic governing equations (4.8) and (4.11), we need to discretize the computational domain. Eulerian methods discretize the domain into grids, whereas Lagrangian-based numerical methods discretize the domain into computational particles. The fluid properties are updated using the particles. There is no connectivity information between the particles.

One popular Lagrangian particle method is smoothed particle hydrodynamics (SPH). Monaghan (68) presented a SPH method for modeling free surface incompressible flows. Free surfaces can be modeled as particles that repel the fluid interface. (81) also studied the possibility of simulating mixing fluids with interface breakup.

In multi-fluid simulation, the general mass conservation equation yields

$$\frac{\partial f}{\partial t} + \nabla \cdot (f\vec{v}) = 0 \tag{4.12}$$

where f is a fluid property such as mass or density. The Lagrangian formulation of equation (4.12) yields

$$\frac{\mathrm{D}f}{\mathrm{D}t} = -f\nabla \cdot \vec{v}. \tag{4.13}$$

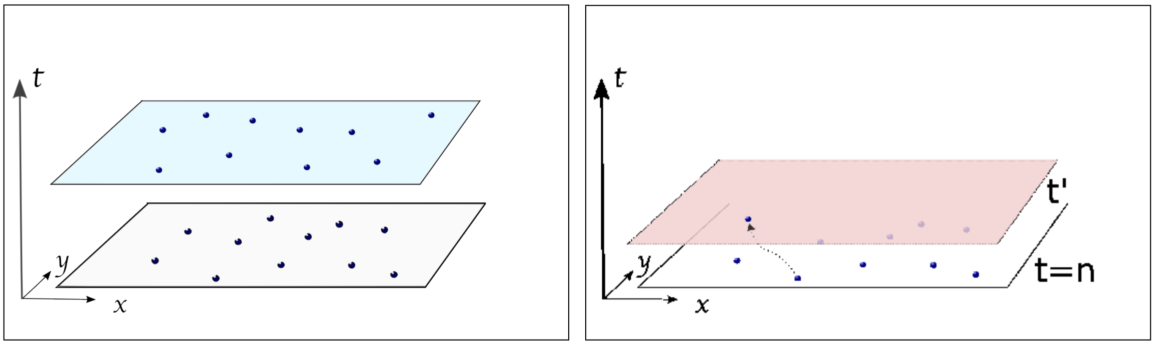
where $-f\nabla \cdot \vec{v}$ is the advected term.

The left term in equation (4.13) is known as the material derivative. Incompressible fluids are divergence-free, i.e.,

$$\nabla \cdot v = 0.$$

Therefore, a change in the material derivative at a material point (the same as a computational particle) yields zero, which is the main concept in Lagrangian particle advection.

In most Lagrangian-based simulations, the particle identifiers are retained throughout the simulation. An initial set of points is injected into the computational domain and advected by the flow (see fig. 4.3b). However, the particle identities are not retained throughout the simulation by some algorithms, such as the finite point set method (100). During each computational time step, the insertion and deletion of particles is conducted to ensure a better discretization quality in the domain (see fig. 4.3a). Therefore, the particles do not correspond between time steps. In this case, a special interpolation scheme is required for interpolation and approximation.



(a) At each time step, a new set of points are initialized

(b) Lagrangian advection on a point

Figure 4.3: Point initialization for Lagrangian method.

4.2.3 Remarks

In grid-based and particle based methods, stability or convergence are not negligible issues. The first order upwind scheme is unstable so it may result in poor convergence. With particle-based methods, the advection term of computational particles can be approximated using finite difference schemes based on a radial basis function. Moving particles must be conducted with care when the velocity deviation is high. A common way of addressing the deviation problem is to apply an adaptive integration length.

In this chapter, we reviewed current methods for material interface reconstruction and tracking analysis. Grid-based and point-based numerical methods can be used to solve the governing equations.

In Chapter 6, we introduce a new interpolation scheme for point-based data that has a random point distribution at each time step.

Chapter 5

Fluid interface instability

[1] Flow simulations, one of the most widely used classic simulation types in physical modeling and visualization, are, for example, no longer limited to simulating just one homogeneous fluid, but have successfully been applied to multi-fluid problems with highly heterogeneous fluid properties. These multi-fluid simulations are capable of accurately modeling fluid interactions and are an important numerical tool to investigate fluid mixing in a number of application areas such as chemical extraction processes, petroleum industry, and combustion systems.

Previous researches presented in chapter 4 are focusing on the extraction of accurate snap-shot representations of multi-fluid interfaces. However, the real challenge in multi-fluid visualization lies in the analysis the flow field in connection with material interface shape and topology. Novel visualization techniques are needed to give insights into interface coherency, interface behavior, and fluid mixing.

In this chapter, we present a novel approach that facilitates visualization and analysis of complex material interface behavior over time, provide a stability or coherency visualization of extracted material interfaces, and for the first time allow a completely consistent display and tracking of material interface evolution. We introduce a material interface stability visualization technique by using parametrization transfer and comparisons between the development of material interface meshes and time-surfaces.

---

[1] This work on material interface instability was published in (72)

In section 5.3, fundamentals on multi-fluid simulations, resulting material interfaces and their reconstruction are presented. In section 5.4, we elaborate our categorization of interface behaviors and further analysis on interface stability. Detailed studies on choice of visualization techniques and user-interactions can be found in section 5.5. Section 5.6 provides an analysis of the developed methods by the use of different numerical examples.

5.1 State of the art

In this section, we will present a literature review on the visualization topics that relate to our approach. Difference between our approach to the existing ones will be demonstrated in section 5.2.

Recently material interface reconstruction work includes simple Line Interface method. It is introduce by Noh and Woodward (71) in 1976. In this method, cells are partitioned with simple axis-aligned lines or planes in order to match and preserve the volume. Later in 1982 Youngs (108) and Rider et al. (84) developed the Piecewise Linear Interface Calculation algorithm. Othert MIR approaches find a smooth and continuous interface based on fractional material data. Such discrete approaches on MIR are done by Bonnell et al. (14), Meredith and Childs (65), and Anderson et al. (3). The former two approaches construct boundaries by calculating intersections between cells in material space with cells that represent the dominance of one material, while the latter uses a volume-adaptive active interface model to generate high-quality boundary meshes. All material interface meshes shown in this work were extracted by a variant of Bonnell's method.

One important characteristic of fluid interfaces is time dependency. In classic vector-field visualization, the dynamics of fluids are captured by integral lines or surfaces. A large body of work has been devoted to address the computation and visual capabilities of these integral surfaces. Following Hultquist's (44) fundamental work on the construction of stream surfaces, van Wijk (103) presented stream surface construction based on implicit surfaces, which can handle irregular surface topology. Computation and visualization of such surfaces has been studied in mesh-based (88) and in point-based settings (87). A recent line of research addresses the computation efficiency and accuracy of integral surfaces (17, 34, 48) by applying mesh adaptivity schemes during extraction.

Texturing of surfaces, adaptive meshes, or flow based texturing has been researched by several authors over the last few years. Texture synthesis models (8, 101, 104) focus on how texture can be aligned to a vector field or given surface shape. However, these models are applied individually to surfaces at each time frame or show loose correspondence only, hence the dynamics of material surfaces are commonly not fully captured. To study how texture evolves along the fluid, van Wijk introduced an image-based flow visualization technique for curved surfaces (104). In the meantime, Neyret (70) and Bargteil et al. (7) proposed texture advection methods to animate a textured surface which is passively advected by the flow field. Texture advection has been applied to iso-surfaces and stream surfaces by Laramee et al. (52, 53).

5.2 The goal of our approach

Our approach combines and develops techniques from three areas: material interface reconstruction (MIR), flow integration, and mesh parametrization or texturing. In the following we give a brief overview of relevant related work in these fields.

In this section, we present the correlation and difference between our approach and the existing ones.

While numerous techniques in vector-field visualization perform domain segmentation by analyzing properties of the underlying flow field, such as vector-field topology or divergence oriented methods, multi-fluid simulations directly imply a separation of the flow field into multiple domains. In many cases it is required to locate, reconstruct or track these interfaces or boundaries between two or more given materials.

In recent years, several different methods to reconstruct material interfaces from output of such multi-fluid simulations have been introduced in the visualization community, most of which can extract snapshot-like interfaces in single simulation time steps (cf. (3, 14, 71)) and approximate interface topology. However, coherent extraction of volume-accurate time-varying material interfaces is still an open problem.

We aim to develop a visualization techniques which handle complex material interface behavior over time. Given reconstructed material interfaces and a time-varying flow field, we make use of time-surface integration to track and perform consistent parametrization of material interface meshes. Several challenges with respect to surface matching and parametrization accuracy are handled by our dynamic remeshing and parametrization seam tracking techniques. The resulting parametrized mesh is used for feature tracking and identification of so-called interface instabilities. These instabilities are visualized as volume rendered particles that detach from the multi-fluid interface. Furthermore, direct user interaction techniques on the parameterized surface are presented that facilitate interactive tracking of interface features over time.

The introduced techniques support the analysis of material interface stability and relation to multi-fluid mixing. Our approach contributes to the visualization community in the following ways:

- Consistent parameterization of material interfaces throughout a complete simulation.
- Visualization of material interface stability and coherency.
- Interactive visual interface tracking.

The remainder of this Chapter is organized as followed. Section 5.3 presents the fundamentals and background information on multi-fluid simulations, resulting material interfaces and their reconstruction. In section 5.4, we define our classification of interface stability types, introduce our algorithm, and illustrate how we handle initial parameterization and advection. Interaction and visualization techniques for the parameterized interfaces and surface instabilities are presented in Section 5.5. An analysis of the developed methods by the use of different numerical examples is performed in Section 5.6.

5.3 Material interfaces

We describe the back ground information of the VOF simulation and MIR techniques. We also prepare ourselves with the definition of interface behaviors, which is the basic of understating surface stability. Material interfaces are given as free surfaces or fluid boundaries in multi-fluid simulations. These simulations and their output can take several forms (39, 50, 102), one class of which, the *volume of fluid* (VOF) ((39)) method is particularly wide spread and used throughout this work. In the following we introduce the necessary background of VOF simulations, material interfaces and their origin and reconstruction.

5.3.1 Volume of fluid simulations

In VOF methods, a fluid identification function is advected during simulation and discretized in the form of a *volume fraction* function f at output. For two fluids F_1 and F_2 in a three-dimensional time-varying simulation such a volume fraction function

$$f : \mathbb{R}^3 \times \mathbb{R} \to [0,1] \tag{5.1}$$

is given as the percentage of fluid F_1 present in a given volume or cell V of the data set. Consequently, an arbitrary cell or volume V of the domain contains a fraction of $f(V)$ of fluid F_1 and a fraction $1 - f(V)$ of fluid F_2. When generalized to multiple fluids, this volume fraction function becomes vector valued. During advection of the fluid identification function, mass of the fluids as well as fluid identity of flow particles is conserved. The VOF method is well-known as a simulation techniques that can handle the occurrence of complex fluid interface behavior including interface topology changes such as bubble break-off (see Figure 5.1).

5.3.2 Interface reconstruction

Material interface reconstruction in multi-fluid simulations is concerned with the extraction of a geometric representation of fluid boundaries. In the context of VOF methods, this requires the processing of all cells or volumes V that are not covered entirely by one fluid, i.e., $f(V)_i \neq 1 \quad \forall i \in \{F_i\}$.

In such a cell the concrete geometric representation of the fluid interface is not unique as illustrated in Figure 5.1. More plausible interface representations can be obtained by incorporating volume fraction data from neighboring cells into the extraction

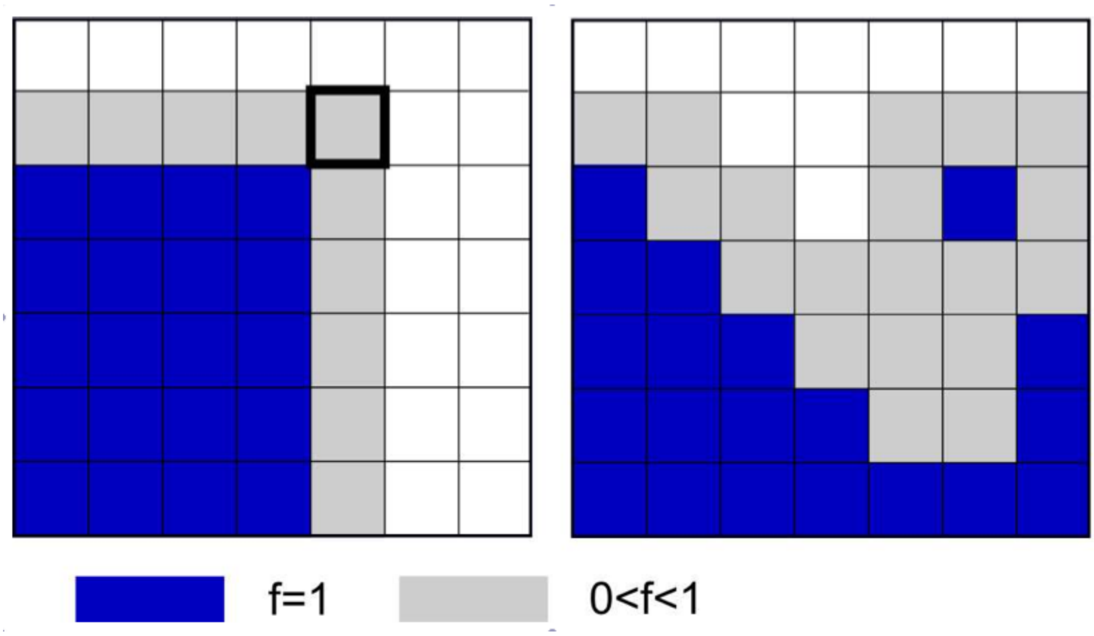


Figure 5.1: Two time steps of a 2D VOF simulation with possible topology change. The interface lies in the region with a volume fraction between 0 and 1.

process. However, neither the shape, nor the topology of the extracted material interfaces is unique for a given non-trivial volume fraction function. Thus, different MIR techniques may extract different surface representations, leading from discontinuous representations that only approximate the given volume fractions (71, 84) to continuous and smooth methods that adhere to the specified volume fraction data (3, 14).

In the remainder of this chapter, we assume that we are given an extracted material interface mesh for every time step of the simulation along with the corresponding flow field. Note that virtually all MIR techniques, even the discontinuous ones, yield a representation that can be converted into a suitable triangulated mesh.

5.3.3 Interface behavior

In addition to topology variation caused by the volume fraction function itself, or resampling, ambiguity of material interface reconstruction can lead to sudden changes in topology over time as well. As a consequence, subsequent material interface representation often exhibit significantly different local topology. This is a stark contrast to other surface representations known in the visualization community, namely integral surfaces such as path- or streak-surfaces, and significantly complicates consistent and coherent visualization of interface surfaces over time. In this chapter, we analyze the behavior of classic material interfaces and time surfaces, making use of the following definitions:

A material interface in a three-dimensional time-varying VOF simulation with n time steps $t_1,\dots,t_n$ consists of an ordered set of material interface surface meshes $\{M_1,\dots,M_n\}$ with $M=\{V,E\}$ being a tessellated surface mesh with vertices $v_i \in V \subset \mathbb{R}^3$ and edges $e_j \in E = \{v_{j1}, v_{j2}\}$. Edges in E delineate individual surface cells, such as triangles or quadrilaterals. A point $x \in \mathbb{R}^3$ is said to lie on M if it is located on the piecewise linear representation of M given by the surface cells. In the following, M_i and M_j with $i \neq j$ are assumed to have no explicit relationship apart from the fact that they were both extracted by the same MIR algorithm (this constraint may be weakened as emphasized in the results section). Thus, material interface meshes of different time steps do not have to show a consistent behavior or be comparable in a meaningful way. This definition allows for arbitrary difference in topology between subsequent material interface meshes.

In contrast to material interfaces, an integral time-surface in a three-dimensional time-varying flow field $g:\mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^3$ consists of individual mesh representations $\{T_1,\dots,T_n\}$ that are created by flow advection of an initial surface. Such a time surface representation is again a tessellated surface mesh with vertices $v_i \in V \subset \mathbb{R}^3$ and edges $e_j \in E = \{v_{j1}, v_{j2}\}$ that delineate individual surface elements. The iterative character of time-surface creation guarantees that subsequent representations possess a shared surface parametrization.

Thus, a parametrization function p with $p: T \rightarrow \mathbb{R}^2$ exists for every T, and the property of temporal parametrization consistency between subsequent time-surface representations T_i and T_{i+1},

$$p_j(s(v_i,t_j)) = p_{j+1}(s(v_i,t_{j+1})) \tag{5.2}$$

holds, with

$$s(x,t) = x + \int_{t_0}^{t} g(s(x,\tau),\tau)d\tau, \quad s(x,t_0) = x \tag{5.3}$$

corresponding to the common definition of integral path-lines. In other words, equations (5.2) and (5.3) ensure that surface parameterization is preserved during advection. Thus, there is a unique correspondence between subsequent time-surface meshes that implies a consistent parametrization but makes global topology changes of time surfaces impossible.

In summary, time-surfaces correspond well to flow behavior but not necessarily material boundaries, since they cannot model topology changes. Material interfaces, on the other hand, while corresponding to snapshots of material interfaces, do not inherently possess a coherent parametrization. Both surface and mesh types can be extracted from VOF data, a fact that allows the combination of both surface properties as demonstrated in the remainder of this work.

5.4 Material surface stability

The definitions given in the previous section allow us to define the notion of material surface stability. The central question we want to answer in this work is: How do time-surfaces and material interfaces correspond and how stable is their behavior with respect to one another? The answer to this question can help evaluate material interface reconstruction methods, track interface features, and highlight interface mixing.

5.4.1 Definition

In the following, a material interface M_i and a time-surface T_i in time step i are said to *correspond*, if the time-surface was created by advection of a material interface M_j, $j <= i$. Analysis of the evolution of corresponding time-surfaces and material interface meshes reveals one of the following behaviors:

1. *Stability*: If there exists a pair of positions x_M and x_T on a material and a corresponding time-surface at time step j with $\|x_M^j - x_T^j\| < \varepsilon$ then there exists a position x_M^{j+1} such that $\|x_M^{j+1} - p_{j+1}^{-1}(p_j(x_T^j))\| < \varepsilon$. Such a point x_M^j is therefore a stable material interface point with respect to the flow field and stays within an ε band of the time-surface. Note that p^{-1} locates a point on the surface in parameter space.

2. *Detachment*: If there exists a pair of positions x_M and x_T with $\|x_M^j - x_T^j\| < \varepsilon$ and there is no position x_M^{j+1} with $\|x_M^{j+1} - p_{j+1}^{-1}(p_j(x_T^j))\| < \varepsilon$. Such a point x_T^j is therefore being detached from the material interface representation by flow advection. That means it leaves an ε band of the material interface.

3. *Attachment*: If there is no such position x_T^{j+1} for a given x_M such that $\|x_M^{j+1} - x_T^{j+1}\| < \varepsilon$. Such a point x_M^{j+1} in the flow field is therefore being attached to the material interface representation by flow advection. That means it enters the ε band of the material interface.

In the following, we denote detachment as material surface instability, whereas stability is used for parametrization transfer. There are numerous causes for attachment or detachment behavior in surface evolution of reconstructed interface meshes. Most frequent causes include natural topology changes in material interfaces, a non-unique extraction process with respect to VOF interface topology (Section 5.3), sampling resolution or strategy of the reconstruction method, and accuracy of the reconstruction with respect to fluid volumes.

5.4.2 Algorithm outline

Assuming a VOF simulation was run and provides extracted material interface meshes for the whole length of a time-varying vector field g, our algorithm to detect and process material surface stability, detachment, and attachment takes the following steps:

1. Parameterize material interface M_0 with an injective function p, select M_0 as current interface M.

2. Use material interface M as seed surface of a time-surface with a corresponding parametrization function p of M.
3. Advect the time-surface to the next time step.
4. Parameterize the next material interface by identifying regions of surface stability.
5. Advect instable regions of the surface through the flow field.
6. Repeat from step 2 with the material interface of the next time step.

In the following sections, we explain these six steps in more detail.

5.4.3 Initial parametrization

Material interfaces of all time steps will serve as seeding structures for time-surfaces as indicated by step 2. Thus, obtaining a correct classification of stable and unstable material interface regions requires the presence of a well-defined parametrization function p for the first material interface M_0. This parametrization is created explicitly for the first material interface mesh and transferred to subsequent material interfaces as described in 5.4.5.

Mesh parametrization is a well-researched area and has led to a number of techniques including patch-based local parametrization, methods related to texture synthesis, and global parametrization techniques based on mesh flattening (12, 92, 101). We prefer a global parametrization scheme over local patch-based schemes as they insure that p is an injective function. Such schemes reduce the amount of parametrization seams and allow interactive modification of the mesh in a shared parameter space as demonstrated in Section 5.5.2.

The global parametrization scheme of our choice cuts the mesh along areas of strong distortion and unwraps the resulting open 2-manifold to the plane by angle based flattening. For this manner we use custom implementations of seam layouting (93) and mesh flattening (92). Note that this parametrization function p is discontinuous along the generated seam edges but minimizes angular distortion during parametrization. Simpler global parametrization strategies such as sphere mapping are applicable as well, but often introduce high distortions, singularities, and complex discontinuities and are frequently non-injective functions.

5.4.4 Interface advection

After parametrization a material interface mesh is used as seeding structure for time surface advection. In the simplest case, we set $T = M$ and advect the surface to the next time step by using an adaptive Runge Kutta integrator of order five (20). In this case mesh connectivity and resolution stays static during advection.

More complex time surface seeding and advection strategies include subsampling of the material interface mesh and performing adaptive time-surface integration (48) to save integration time, as described later in this paper. Hereby two constraints on surface simplification have to be fulfilled. First, prevalent surface features such as edges, ridges, and valleys must not be missed by the merging strategy, and secondly, regions where the parametrization function behaves non-linearly must not be merged during the simplification process.

5.4.5 Stability classification

The parametrization of the advected time-surface has to be transferred to the material interface mesh in the next time step to facilitate stability analysis and consistent interface parametrization. For this matter we perform two-sided mesh matching.

Given a distance threshold ε and a vertex v on the material interface mesh M in the next time step, we loop through all cells, edges, and vertices of the advected time surface T to find the closest interpolated position on T. This element search is locally constrained by first voxelizing T and its elements into a uniform grid with cell sizes corresponding to ε. If the closest position $x \in T$ satisfies $||x - v|| < \varepsilon$, parameter values of x as obtained by barycentric interpolation on cells of T are assigned to v, declaring it as part of a stable region. If there is no such closest position, v represents an attachment point. Since attachment points have not been part of the material interface previously, they are assigned distinct parameter values outside of the mesh's regular parameter space that identify them as attachment points.

The second matching is performed by swapping the roles of T and M. For every vertex on T we find a closest point $x \in M$. If $\|x - v\| < \varepsilon$, v is part of a stable region. If there is no such x, v represents a detachment point, or a material surface instability.

There are certain cases especially during interface merging events, e.g., merging of a bubbles, where several non-neighboring positions $x \in T$ satisfy the distance constraint $\|x - v\| < \varepsilon$. There are several options to handle these cases. a) Mark v as possessing multiple positions in parameter space, b) assign the parameter value of the closer position to v, or c) store v as non-tracked, unparameterized vertex with parameter values outside of the valid parameter space. The examples shown in the results section make use of option a) for computational simplicity.

5.4.6 Instability tracking

Instability points as identified in the previous section indicate regions, where subsequent extracted material interface meshes are not coherent with fluid motion, e.g., in regions with strong topological changes. Tracking these detachment regions over time can be used to evaluate consistency of the extracted material interfaces, analyze flow divergence and quantify mixing, as these instabilities used to mark borders between different liquids.

To this end we seed instability particles on these regions and advect them through the complete simulation. Such instability particles store information about the time step of detachment as well as their last valid parameter values for visualization purposes. In the examples used, we seed particles at instable vertices and adjacent triangle center positions.

5.4.7 Improvements and optimizations

The steps detailed in the previous sections describe a full run of our algorithm. There are, however, several additional optimization and improvement considerations that we detail in the following.

5.4.7.1 Seam treatment

Global parametrization schemes introduce seams when applied to closed meshes. If these seams are not handled specifically during parametrization transfer, severe visual artifacts can occur that worsen as the simulation progresses.

Given a cell c of material interface mesh M, an edge $e = (v_1, v_2)$ of c contains a discontinuity of the parametrization function if vertices v_1 and v_2 map to opposing sides of a parametrization seam. If mesh parametrization is used for texturing, the cell is incorrectly mapped with a large part of the parametrization texture. Figure 5.4 shows an occurrence of this phenomenon.

Such an invalid cell or triangle can be identified by analyzing its properties in parameter space: It either has a flipped normal in parameter space, extremely large edges in parameter space, or causes strong local compression of parameter space when compared to the last time step. Several triangles that fulfill all of these properties are visible in Figure 5.4.

Since the correct projection into parameter space is unknown for an invalid triangle and cannot be computed analytically, we employ an iterative edge-bisection technique to create sub-triangles that match the parametrization seam. For an edge $e = (v_1, v_2)$ this bisection techniques works as described by the following pseudocode (see also Figure 5.5).

Control over accuracy of seam approximation is given by choice of δ. In practice we chose δ around $2e-4$. Lower values do not slow seam approximation significantly, but produced no noticeable difference due to small triangle sizes in the used test data sets. After one run of the algorithm, we have a position v on the edge that approximates the seam location and two locations in parameter space $p(w_1)$ and $p(w_2)$ that lie on opposing sides of the parametrization seam. Once this algorithm is run for each edge of the invalid triangle, we are left with one, two, or three intersected edges. The affected triangle is remeshed by splitting edges at p. Incorporation of $p(w_1)$ and $p(w_2)$ into the re-meshing process ensures that all vertices of the new triangles are located on the same sides of the seam. Figure 5.6 shows the triangle splitting and re-parametrization process. In the special case of three cuts, parametrization of one of the new sub-triangles is extrapolated by using laws of cosine, as common in mesh unwrapping (92).

Algorithm 2 Pseudocode for edge bisection

```
$w_1 \leftarrow v_1, w_2 \leftarrow v_2$
while $\frac{\|w_1 - w_2\|}{\|v_1 - v_2\|} > \delta$ do
    $v \leftarrow (w_1 + w_2) \cdot 0.5$
    if edgeIsInvalid($p(v)$,$p(w_1)$) then
        $w_2 \leftarrow v$
    else if edgeIsInvalid($p(v)$,$p(w_2)$) then
        $w_1 \leftarrow v$
    end if
end while
```

If more than one seam crosses a triangle edge, the triangle is split and the procedure is applied to all sub-triangles.

5.4.7.2 Parametrization accuracy

During parametrization transfer from the time surface to the material interface mesh, significant parametrization details can be lost due to resampling of parameter space on a mesh vertex level. This becomes especially critical if an adaptive time-surface was used for advection and interface mesh vertices show very low correspondence in position and density. If an accurate representation of small scale parametrization features is desired, the material interface mesh has to be retriangulated during the parametrization transfer process. That means vertices of the time-surface are not only used to transfer surface parametrization, but serve to remesh their closest triangle of the material interface mesh as well. This can be seen as projecting the time-surface connectivity onto the material interface mesh. The major drawback of this retriangulation step is increased resolution of subsequent time-surfaces and therefore largely increased advection times.

5.4.7.3 Performance considerations

Analysis of algorithm performance in Section 5.6 indicates that time-surface advection take up the largest part of overall computation times. In the following we briefly discuss optimization strategies to reduce computational overhead. Note that it is sufficient to keep two subsequent time steps in memory for integration, since there is no other complex interaction along the time-axis.

Parallelization Particle and time-surface advection is directly parallelizable on a shared memory machine by distributing path-line computation to different threads. If a non-adaptive integration scheme is sufficient, advection can be performed on the GPU (17). Closest-point identification for parametrization transfer is parallelizable on a per-vertex level as well.

Mesh subsampling An optimization strategy closely related to parametrization accuracy is mesh subsampling as mentioned earlier. We simplify a material interface mesh by removing a mesh vertex v if the following constraints hold for v and its projection point v_{proj} on the simplified mesh

$$\begin{gathered} curv(v) < \delta_1 \\ \|p(v) - p(v_{proj})\| < \delta_2, \end{gathered}$$

where $p(v_{proj})$ is obtained by barycentric interpolation on the cell of the simplified mesh that contains v_{proj}. By constraining the deviation between parametrizations of the full and the simplified mesh, we ensure no significant parametrization detail is lost. If additionally the maximal resulting mesh edge size is constrained, the mentioned constraints guarantee that neither prevalent surface features, i.e., features with large mesh curvature, are missed, nor regions with non-linear parametrization behavior are merged. Feature preservation is controlled by choice of δ_1, whereas parametrization details are preserved by δ_2. These two parameters influence computation times and parametrization accuracy as demonstrated in the results section. Note that strong mesh simplification is best used with subsequent adaptive time-surface integration to insure no important flow features are missed. To improve coherency with high-resolution surface advection, the surface can be upsampled during instability classification.

5.5 Visualization and interaction

This section serves to define visualization challenges that come with material interface stability visualization and detail the proposed visualization techniques to address these challenges. Furthermore, we detail a direct interaction method that allows user-specified visual feature tracking by direct surface drawing.

5.5.1 Visualization

The data to be visualized consists of a set of material interface meshes with according parametrization and a set of detachment points that indicate material surface incoherencies and mixing regions. For visibility reasons neither detachment regions nor material surfaces should be rendered in an opaque manner, thus more sophisticated transparency rendering techniques are required.

For simple material interface mesh visualization we make use of shaders for depth-peeling, texturing, and per pixel Phong shading. Attachment regions are identified by large texture coordinate offsets and rendered either in black or fully transparent. Transparency of the surface is modulated by normal direction relative to the viewer to allow a clear look at interior structures.

Visualization of the instability particle set could be performed in numerous ways including splatting, rasterization, particle geometry, or point set visualization (45, 47). We opt for for two alternatives: Fast density-based volume rendering performed by counting particles present in cells of a low resolution three-dimensional texture. This texture is subsequently visualized by slicing-based volume rendering. Alternatively we use particle splatting if individual particle properties are to be shown. In contrast to the splatting technique, density map visualization requires no particle sorting for correct transparency rendering and rendering performance is independent of the number of particles. A trade-off, however, is the lack of individual particle information present in density maps.

A challenge that arises from our choice of visualization is the combination of two different transparency techniques, namely volume-rendering on proxy geometry for particles and depth-peeling for surfaces. We combine both approaches by employing the depth-buffer layers created during depth-peeling to perform depth-based interval clipping. First, layers of the material interface mesh are rendered to off-screen buffers b_i by the application of standard depth-peeling techniques. The final image is then composed by traversing the stack in back to front order: We blend the color of layer b_i to the frame buffer, followed by drawing the transparent proxy geometry (e.g., slices for volume rendering or textured splats) with parts that exceed the depth range specified by depth-buffers b_i and b_{i+1} being clipped in a fragment shader. Depth buffers b_0 and b_n represent the near and far clipping plane.

5.5.2 Interactive surface drawing

As opposed to classic texture synthesis methods, the availability of a consistent global parametrization of material interface meshes over time supports tracking of feature development. Visual tracking of features can be aided significantly by facilitating interactive modification of a parametrization texture directly on the mesh.

We give the user the capability to draw on the mesh and track the evolution of this drawing over time to observe stretching, deformation, and movement in general. Our implementation follows standard picking procedures. When the user clicks or drags over the mesh, we redraw a small part of the surface around the mouse position by frame-buffer scissoring and implement color based per-triangle picking. The exact position on the triangle is then found by ray-triangle intersection in three-space, allowing the computation of selection position in parameter space by barycentric interpolation on the selected triangle. The parametrization texture is then updated accordingly. In the case of line drawing seam discontinuities have to be resolved by mesh seam treatment as proposed in Section 5.4.7.1

5.6 Results and discussion

We have applied our algorithm to three time-varying three-dimensional VOF data sets and present the obtained results in the following. All data sets were simulated with the OpenFOAM simulation toolset (1) and consist of between 63 and 126 time steps. Simulated phenomena include a dam-break scenario, a fluid-in-fluid drop, and a classic

Rayleigh-Taylor instability. Material interface meshes are extracted with a marching-cubes variant of the method of Bonnell et al. (14) and possess averages of between 14000 and 60000 vertices per time step. Note that, in theory, comparison of different extraction methods (even for subsequent time-steps) is possible but lies beyond the scope of this paper due to page limits.

- Dam break This data set consists of 126 time steps, modeling the first 2.5 seconds of a dam break by letting an unconstrained column of fluid stream over an undersized wall-like obstacle. The extracted material interfaces possess two interesting regions which are highlighted by our parametrization and visualization technique. A large part of the fluid describes a slow and relatively consistent dropping motion, whereas a fast front with complex topology changes is observed past the wall-like obstacle. In later time steps, backflow and splashing affects almost all parts of the fluid interface. Corresponding results are shown in fig. 5.7.
- Fluid drop A 2 seconds fluid droplet scenario is represented by 102 time steps. In this simulation an accelerating fluid column is dropped into a slightly lighter liquid, producing a mushroom or bullet-impact like deformation of the fluid column. The initial phase shows an acceleration and deformation of the fluid column, followed by a second phase, the impact of the fluid column with the data set boundary and subsequent rapid topology changes (see fig. 5.8).
- Rayleigh-Taylor instability The mixing of two fluids, the heavier one on top of the lighter one, is simulated for around 1.6 seconds and 63 time steps in this three-dimensional Rayleigh-Taylor instability simulation. The interface exhibits almost uniform deformation in the beginning and is soon governed by the forming of Rayleigh-Taylor fingers (see fig. 5.9).

5.6.1 Results and analysis

Representative snap-shots of all three simulations are shown in Figure 5.7a, 5.8a and 5.9a . All simulations use a gridded color gradient as parametrization texture to support visualization of stretching and rotation as well as visual tracking of interface regions. Observed stretching together with unique coloring in the dam-break and fluid drop scenarios allows in-depth analysis of shape and topology changes as the interface evolves over time that are impossible if no consistent parametrization is available for the interface. Such stretching regions are seen to be especially prevalent around the fronts of the dam-break and fluid drop scenarios. Density-based instability visualization highlights regions of the flow domain that detached from a material interface in prior time steps and are potential mixing regions of the two fluids in a high resolution VOF simulation of the same phenomenon. This facilitates simulation analysis and joint reconstruction error-analysis. A comparison between instability visualization in the dam-break and Rayleigh-Taylor simulations reveals how instabilities in the dam-break scenario are largely caused around the fast advancing front with increased bubble break-off and turbulence, whereas slow Rayleigh-Taylor evolution creates a more even distribution of surface detachment.

A closer look at instability visualization is given in Figure 5.10 for the fluid-drop scenario, where the surface is affected by rapid topology changes in later time steps. No individual instability particle properties are visible in the computationally simpler density based visualization. Particle splatting on the other hand allows depiction of particle properties such as parametrization value in the form of color, indicating where a particle detached from the material interface, thus providing the possibility to backtrack surface instabilities. Material interfaces, density map, and splatted volume particles are blended together correctly. It is interesting to note how instabilities caused around areas with rapid topology changes propagate across the data set in later time steps.

While topology changes, detachment, stability, and attachment are visible in this direct form of visualization, consistent material interface parametrization allows for a visualization in parameter space as well, which allows in-depth two-dimensional analysis of surface behavior. Figure 5.14 shows such a rendering of multiple time steps of a flattened material interface mesh for the dam-break example. Surface stretching and compression can be observed, as well as surface detachment and topology changes in the form of holes or splitting of the parameterized mesh, facilitating a 2D time-varying topology analysis.

Data	Full Detail	Medium Detail	Low Detail
Dam	134	113	93
Drop	139	119	117
R-T	142	135	128

Table 5.1: Total run-times in minutes for surface advection, stability classification, and instability advection. Medium and low detail computations include time for surface simplification.

A central advantage of consistent and injective parametrization is the possibility for consistent parameter space operations, such as interactive interface highlighting. Examples of such user interaction are shown in Figure 5.15 and Figure 5.16 where the development of user drawings created directly on the mesh can be observed. As shown, highlighting of interesting surface features such as bubbles or holes and feature tracking is made possible by user interaction. When particle splatting is used for visualization, drawings influence the color of instability particles as well, thus allowing forward and backward tracking of surface instabilities.

The performance of our method is heavily dependent on the speed of flow field evaluation. Table 5.1 summarizes run-times of our algorithm on a 64bit Intel Core i7 at 2.2 Ghz with 8 GB of memory. The measurements were performed for full detail time surfaces and two levels of interface simplification. Medium and low detail representations contained between 80% and 40% of the original mesh vertices and were obtained by increasing δ_1 and δ_2. Time spent for parametrization transfer and seam treatment amounts to few seconds per time-step of the simulations. It is notable how the Rayleigh-Taylor instability simulation does not benefit as strongly from simplification, as the other simulations do, since high mesh-complexity avoids the removal of vertices in complex flow regions. Similarly, the high number of instability particles in the fluid drop scenario reduce the impact of mesh simplification after an initially high drop. Figure 5.12 shows visualizations of the dam break data set created with full detail and simplified mesh advection. Note how the chosen δ_2 limits the loss of medium scale parametrization details in flat areas, while small scale details such as local rotations are lost.

A characteristic of our algorithm is its dependability on discrete interface representations. While this is a common computer-science problem, where discretization and sub-sampling is often necessary from computational and representational points-of-view, our method facilitates level-of-detail approaches and can theoretically work on arbitrarily highly resolved meshes. From a computational point of view, used resolutions should stay within reasonable bounds of the resolution provided by the VOF simulation data.

5.7 Conclusion

In this chapter, we introduced a method for material interface stability visualization by using parametrization transfer and comparisons between the development of material interface meshes and time-surfaces. The resulting visualizations allow the distinction between detachment, attachment, and stable regions of material interfaces in time-varying VOF data and support visual analysis of material mixing. Furthermore, this methods is suitable to evaluate the consistency of different MIR methods and can also handle multi-fluid scenarios.

Possible extensions to the addressed method includes adaptive parametrization textures that refine with mesh stretching.

Textures and volume rendering are the main visualization techniques involved in this chapter for studying fluid interface. In the next chapter, we will investigate fluid interface from another perspective– topology.

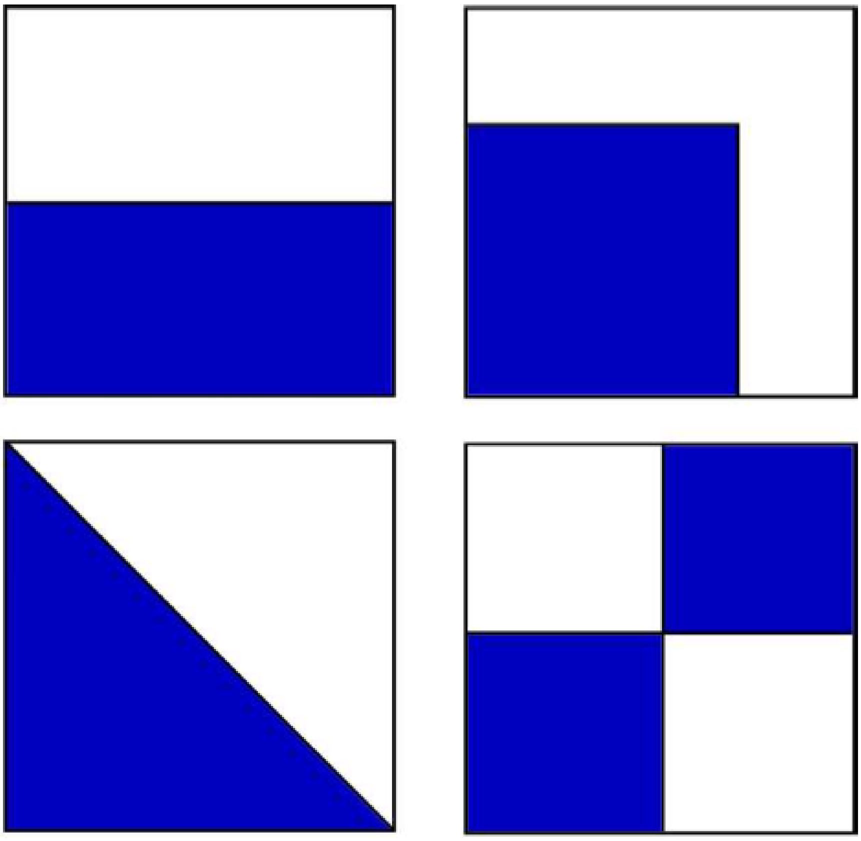


Figure 5.2: Four reconstructed interfaces for the highlighted cell with $f = 0.5$. One fluid is shown in blue, the other in white. All four extracted interfaces are exact with respect to reconstructed fluid volumes. An infinite number of other correct reconstructions exists, leaving the true interface shape and topology unknown.

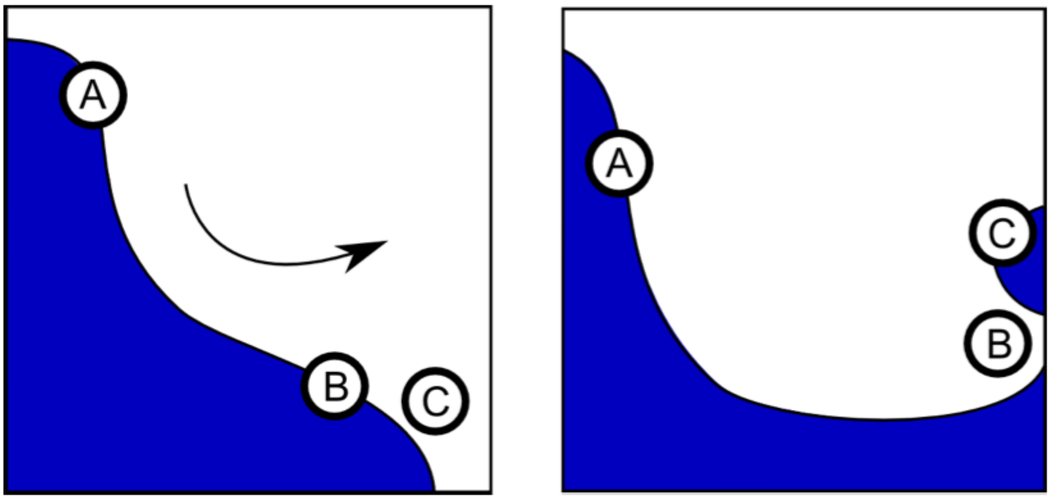


Figure 5.3: Illustration of two consecutive 2D material interfaces. The three points A, B, C are representatives of stable, detaching, and attaching flow particles. A remains on the interface and can be used for parametrization. B detaches from the interface, whereas C attaches to the interface in the second time step.

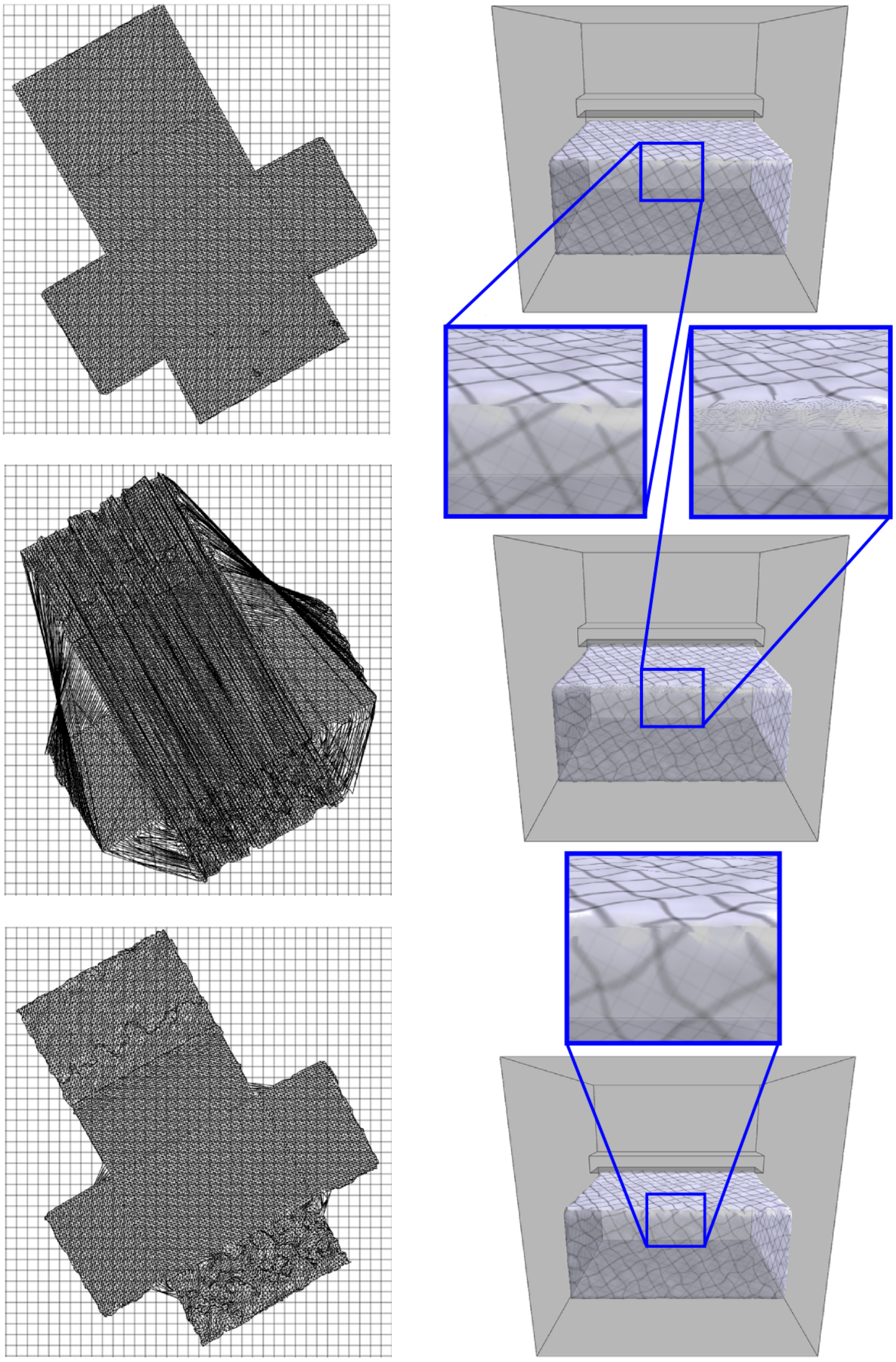


Figure 5.4: Left column: Images of flattened material interface mesh before advection (top) and after advection without and with seam treatment. Parametrization texture is shown in the background. Large triangles across parameter space indicate invalid seam treatment. Right column: Material interface mesh before advection (top) and after advection without and with seam treatment. Close-ups of parametrization seams reveal artifacts if discontinuities are not respected during the parametrization process.

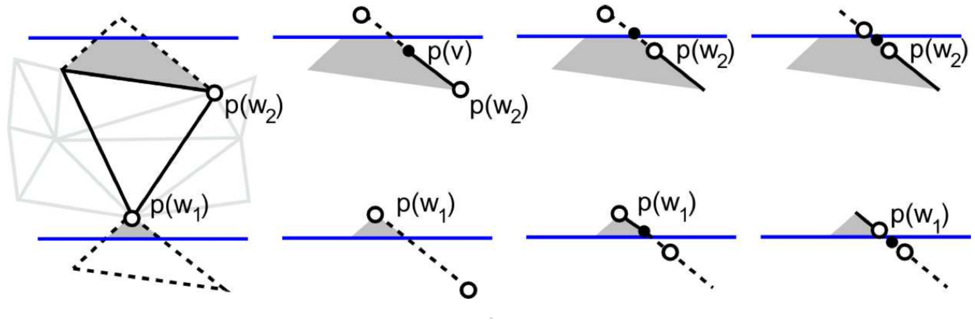


Figure 5.5: Left: A triangle (black outline) is stretched incorrectly over a large region of parameter space, as it crosses the periodic boundary shown in blue. From left to right: Edge bisection and subsequent point-location in parameter space finds intersections between triangle edges and parametrization seams that allow retriangulation of the mesh (gray polygons).

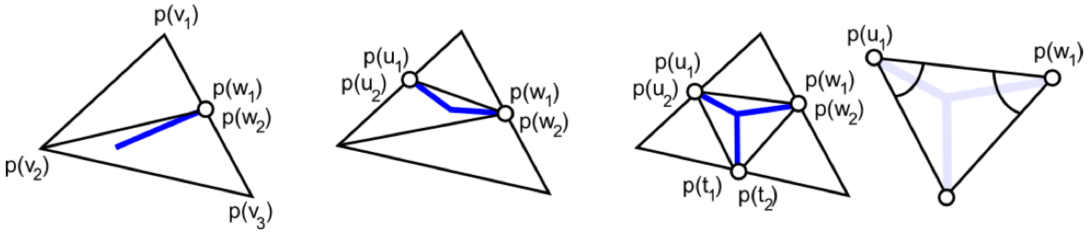
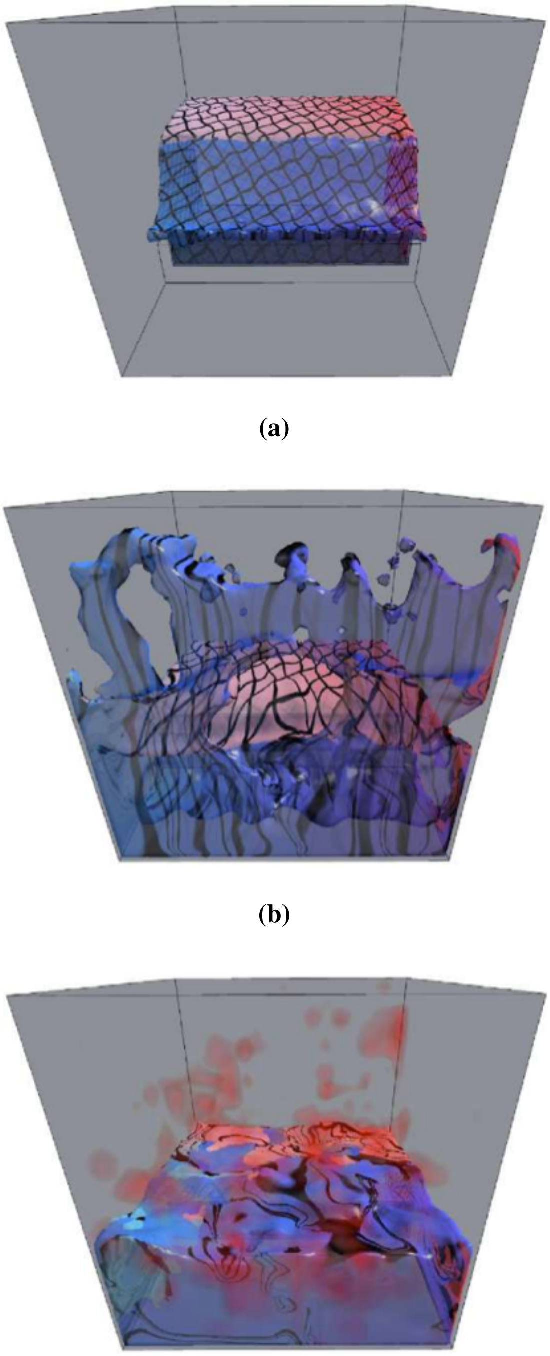


Figure 5.6: Each edge of a triangle may be cut by a parametrization seam. Splitting and reparametrization of the new triangles is straight-forward for the one and two cuts cases (examples of seam locations shown in blue). In the three-cut case shown on the right, we compute parameter values of the third vertex of the middle triangle by using known parameter values of the remaining two vertices together with angle informations in the triangle.

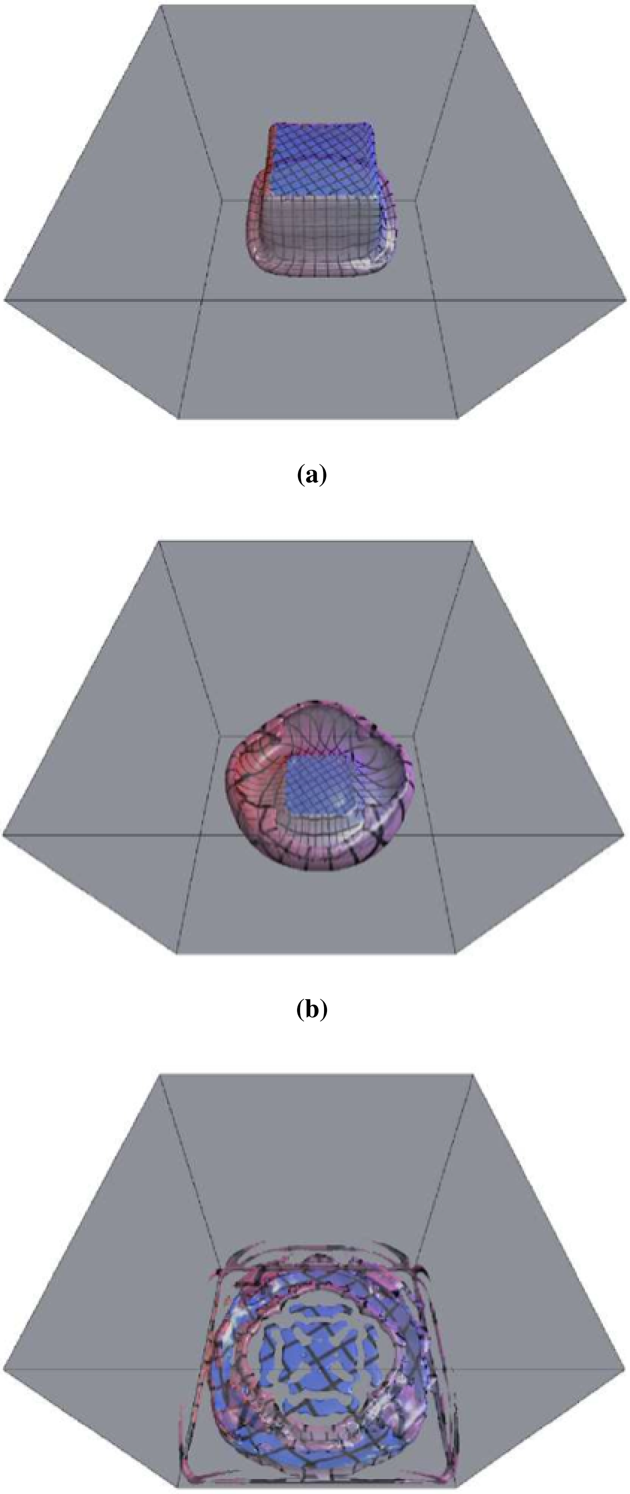


(a)

(b)

(c)

Figure 5.7: Dam-break simulation with volume-rendered instability density in the last frame

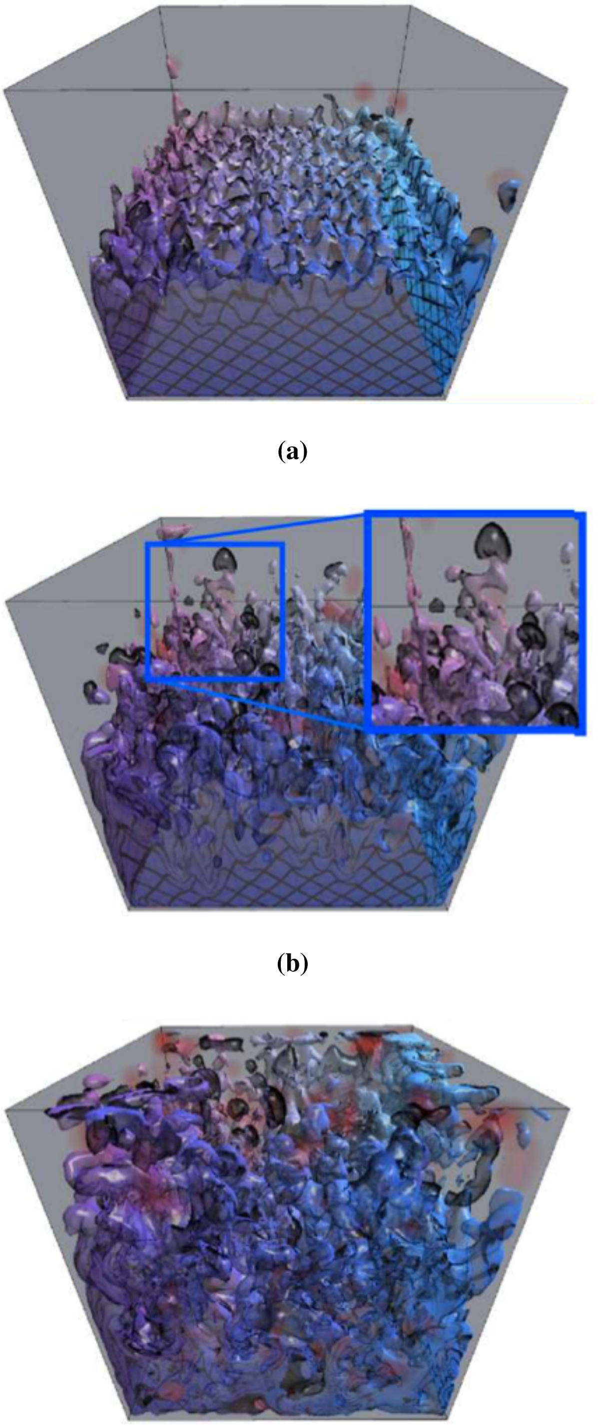


(a)

(b)

(c)

Figure 5.8: Fluid-drop simulation before, immediately before and shortly after impact with the data set boundary rendered without instability

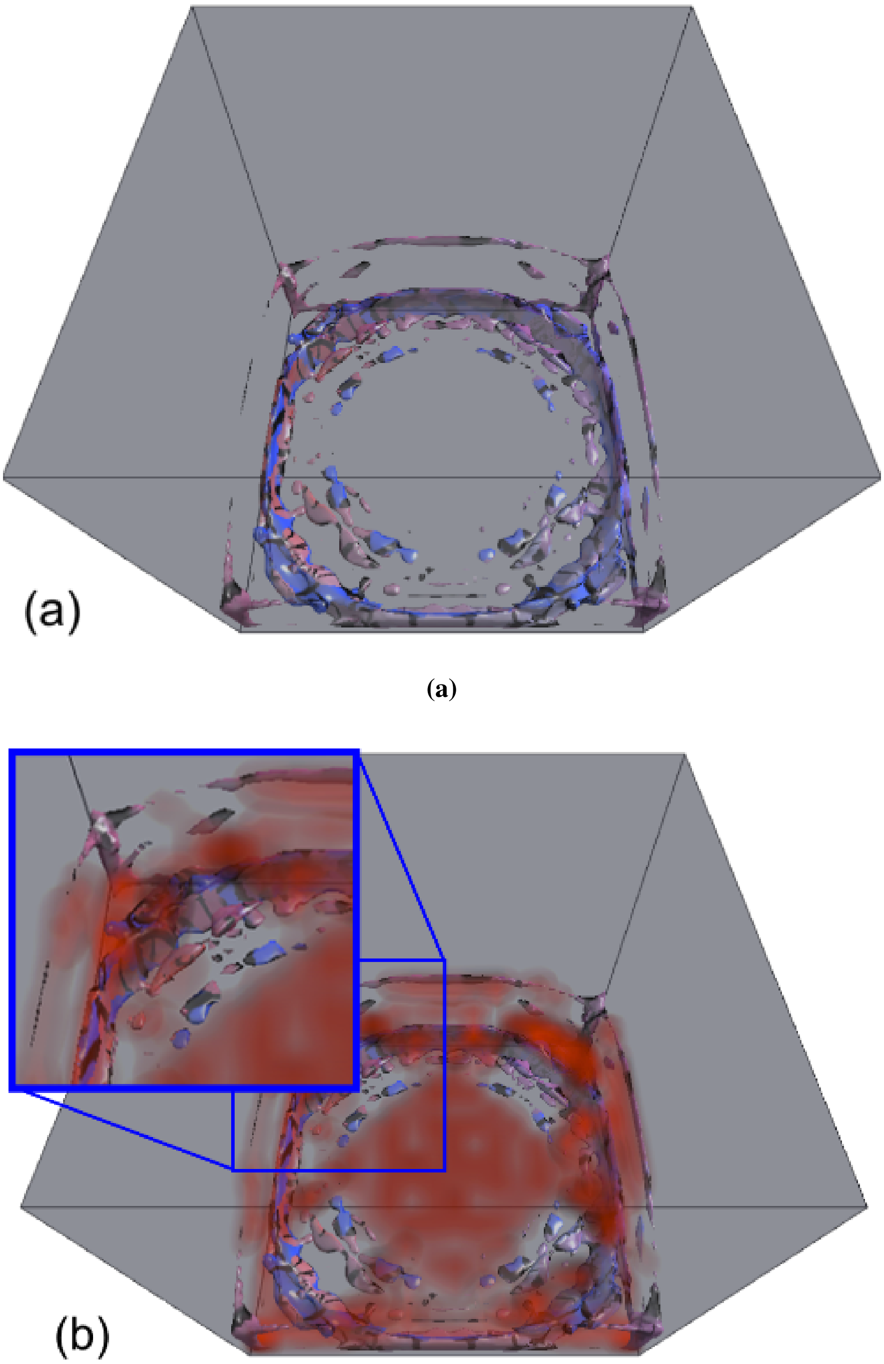


(a)

(b)

(c)

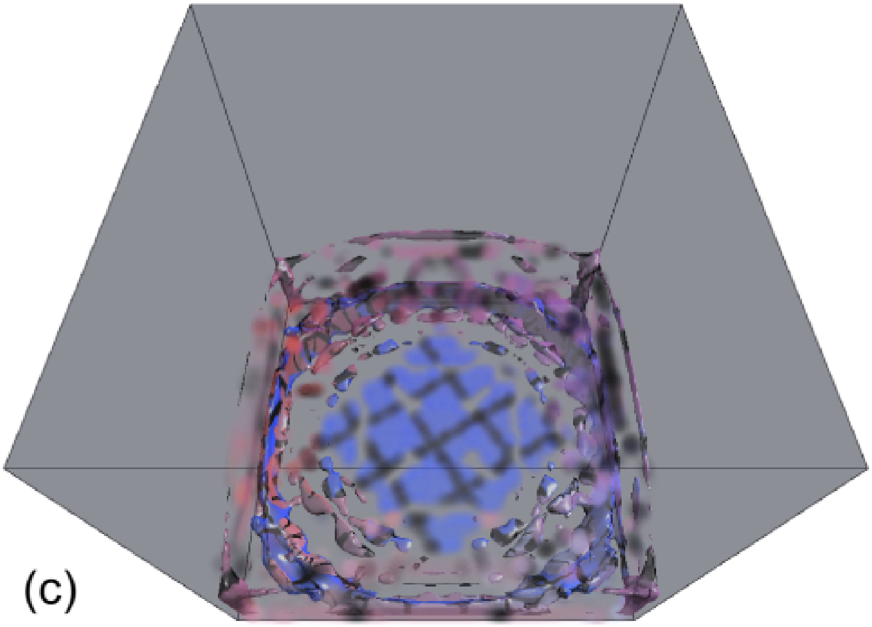
Figure 5.9: Rayleigh-Taylor Instability visualization with density based instability visualization. A close-up at several Rayleigh-Taylor fingers reveals stable (consistent texturing), detachment (red detachment particles), and attachment (no parameterization) situations.



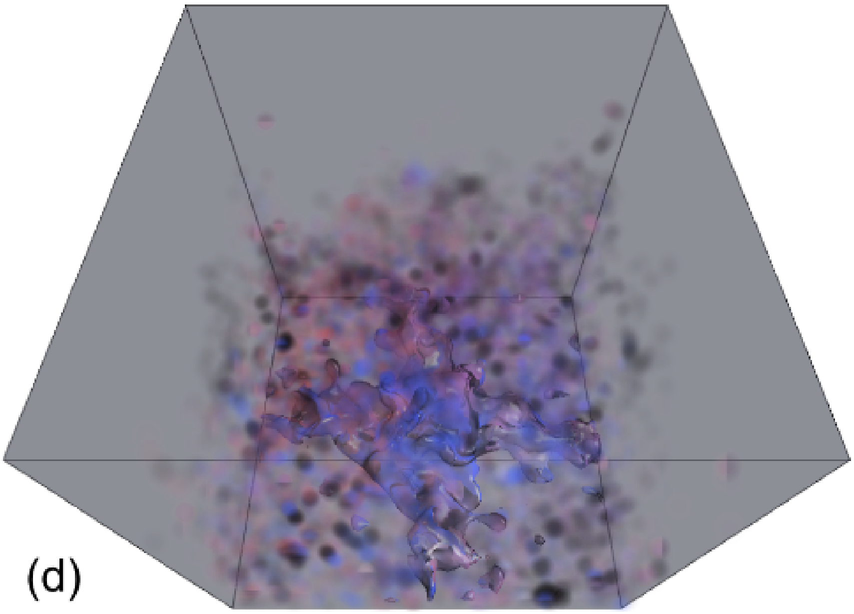
(a)

(b)

Figure 5.10: (a) Visualization of the fluid-drop interface after impact, (b) with density-based visualization.

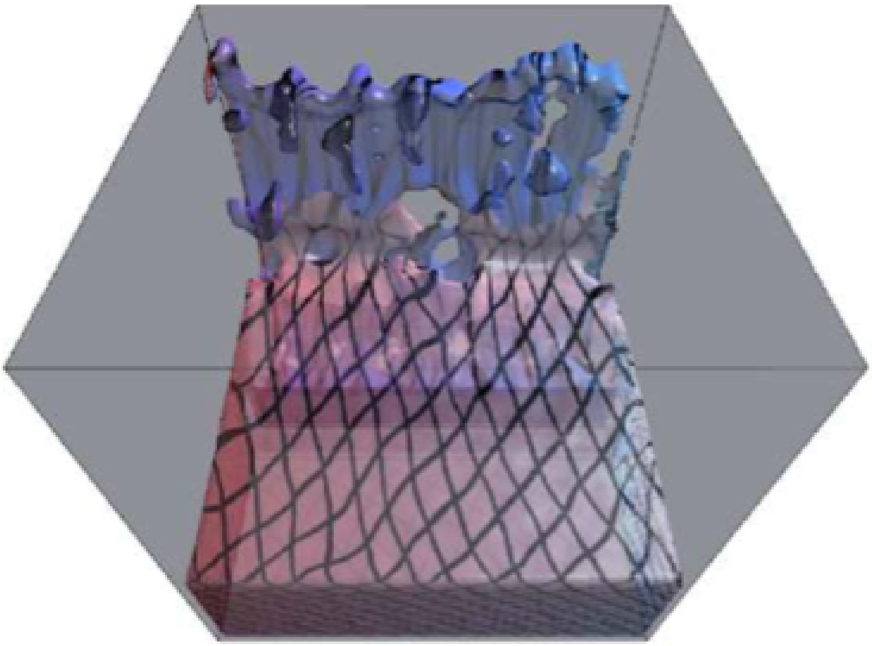


(a)

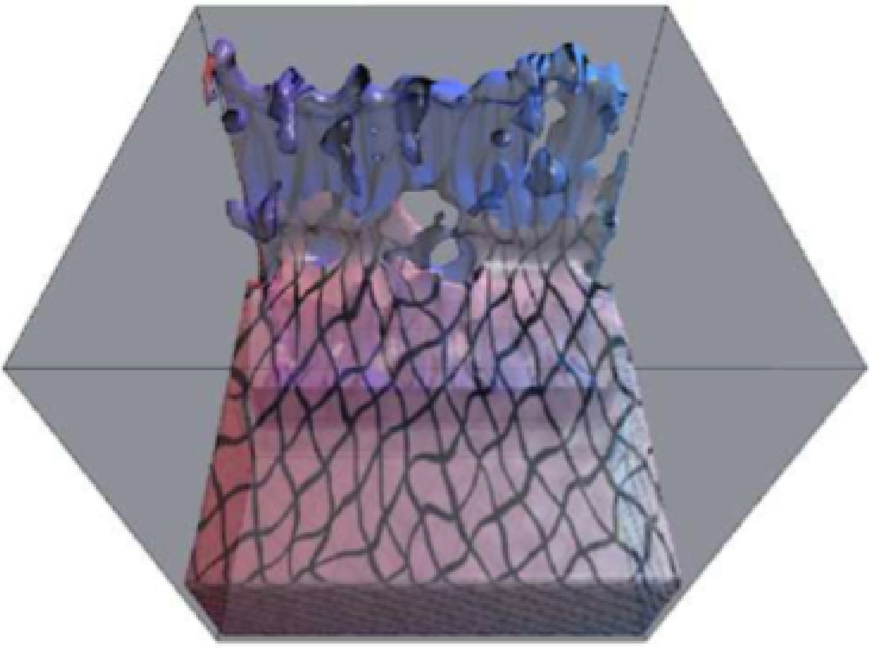


(b)

Figure 5.11: (c) splatting. (d) shows a later time step with instability splatting.



(a)



(b)

Figure 5.12: Full resolution and parametrization transfer with simplified low resolution time surface.

Figure 5.13: The contrast -enhansed image illustrate approximation accuracy of low resolution time surface.

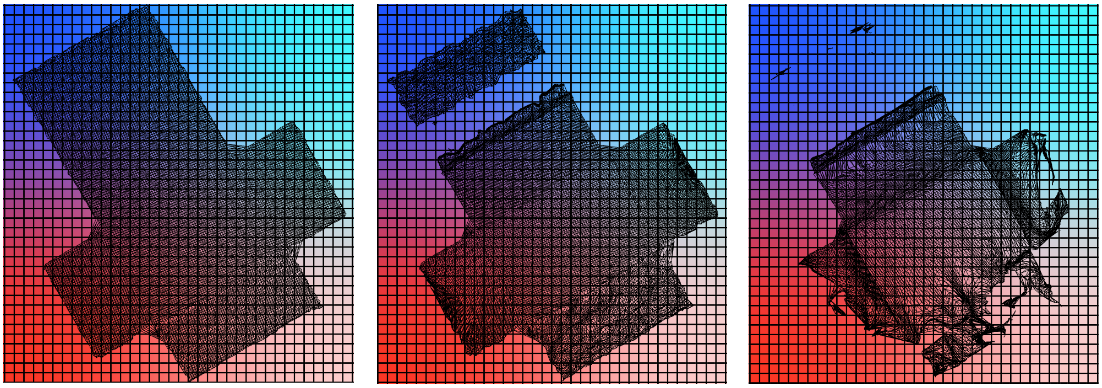


Figure 5.14: The flattened material interface meshes in the foreground of a parametrization texture for three different time steps. Detachment causes topology changes in the parameter space of individual material interfaces. Parameter space seams are treated correctly by the proposed method.

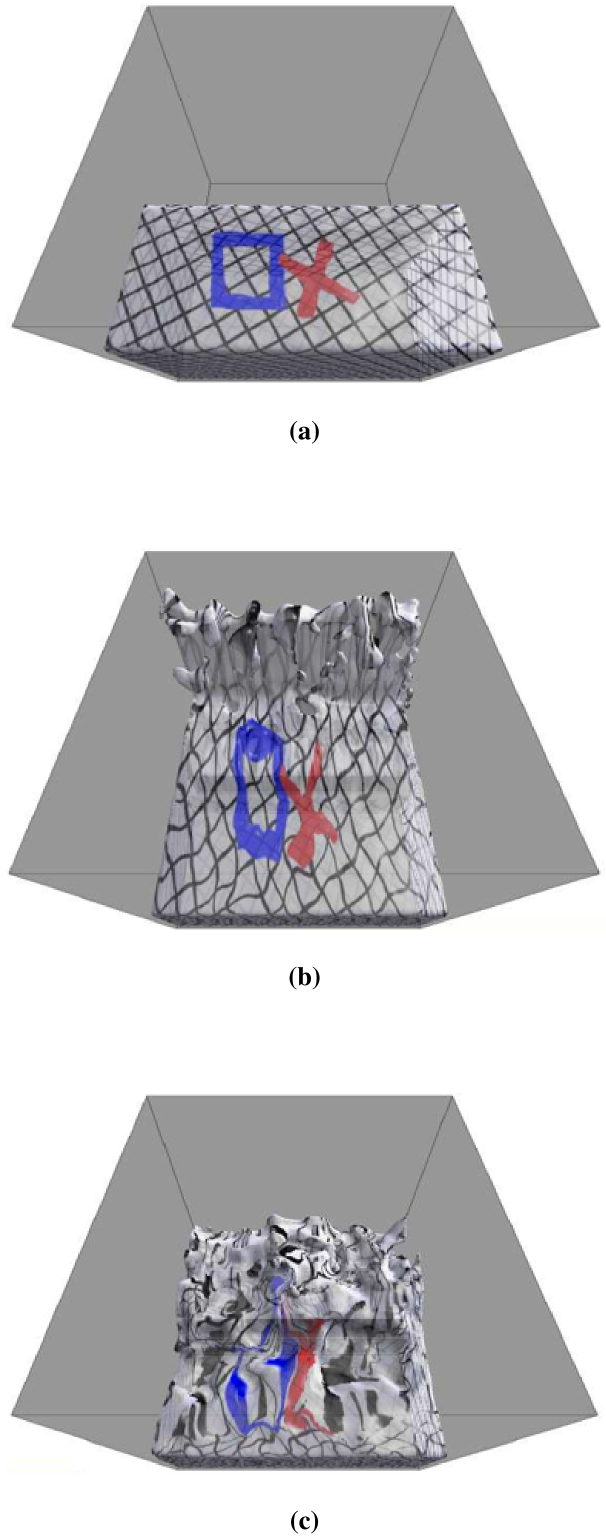
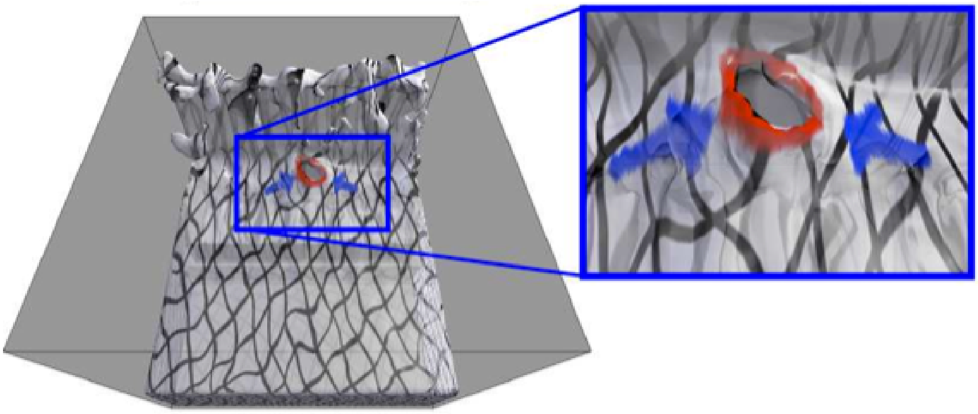
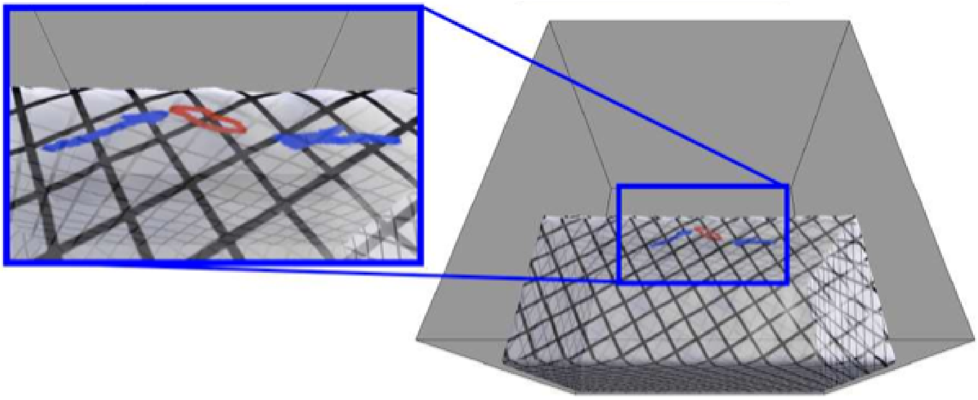


Figure 5.15: Interactive on-mesh drawing for feature tracking: User-made drawing on the initial material interface mesh deforms strongly over time.



(a)



(b)

Figure 5.16: Full resolution and parametrization transfer with simplified low resolution time surface. the contrast enhansed image illustrate approximation accuracy of low resolution time surface.

Chapter 6

The topology-based method

[1]The previous chapter focused on the analysis of fluid interfaces using texture advection. We described texture-based surface visualization using different classes of interface stability. Two interesting classes of interface stability are detachment and attachment. In multi-phase fluids, especially bubbly flows, these two cases are highly related to bubble breakage and collision.

In this chapter, we will study the interface behavior of multi-fluids from the new perspective of interface topology. We treat the fluid density as a scalar function. Investigating the topological structure of the level sets of scalar functions gives us a deeper understanding of the dynamic behavior of fluid interfaces.

We present a topology-based approach for tracking any splitting and merging events in these regions using Reeb graphs. Time is used as the third dimension for the 2D point-based simulation data. The low time resolution of the input dataset means that a physics-based interpolation scheme is required to improve the accuracy of our proposed topology tracking method. The model used for interpolation produces a smooth time-dependent density field after applying Lagrangian-based advection to the given simulated point cloud data, which conforms to the physical laws of flow evolution. The spatial and temporal locations of bifurcations and merging events can be readily

---

[1] This part of the work was published in our paper (23)

identified using the Reeb graph, which supports the in depth analysis of the extraction process.

6.1 Introduction

Multi-fluid interface studies are a growing topic in a variety of application areas. Liquid-liquid extraction is a widely used process in chemical engineering, where the molecules in one liquid are separated by mixing it with a finely dispersed solvent (30). Numerical simulations can model this process as a multi-fluid flow with two liquid phases. The preferred output of these simulations is the predicted evolution of the density distribution for each of the phases, which indicates how well the two liquids are mixed. The density of the fluid phase denotes the fraction of the space occupied by the fluid. During liquid-liquid extraction, a finely dispersed solvent produced low density values for the second phase in large regions of the dataset. FLUENT, OpenFOAM, and FPM are popular CFD solvers used for simulating two phase fluids. FPM was used during the current study, which produces scattered point-sets that possess velocity and density information.

In general, three types of multi-phase fluid models are used to tackle different flow regimes. VOF modeling focuses on tracking the interface of two fluids during slug and surface flow. Eulerian multi-phase modeling deals with the heat and momentum transfer between phases based on discrete phase modeling of the mixture. The final type of modeling is used widely for simulating bubbly flows and slurry flows, such as the fluid mixture found in a bubble column reactor used for liquid-liquid extraction. However, the traditional approaches used for capturing fluid material boundaries cannot cope when multiple regions in a cell are occupied by different fluids, as is the case with finely dispersed liquids. Thus, a higher level visualization technique is required to understand these flow density distributions.

We present a topology-based approach for studying the volume fraction field of an arbitrarily distributed computational point set. The simulation data we considered was a two-dimensional, time-variable fluid field, which was discretized as particles with their associated density and velocity values. The low time resolution of this reference dataset complicates the tracking of material boundaries because a nonphysically-based interpolation scheme can produce an incorrect topology. Thus, we used a physically-based interpolation scheme to improve correspondence between the time steps of the simulation, which facilitated more robust feature extraction.

The major goal of this chapter is to develop plausible and practical interpolation schemes for point-based multi-fluid density data and to characterize the fluid interface behavior using Reeb graphs. To identify interesting time intervals containing regions that are densely occupied by a certain phase of fluid, we first define these regions using a specific *level set* of the density field. Using a physically-based interpolation scheme, we compute a time-continuous density field when resampling the grid points. Finally, we perform a topological analysis of the extracted time-variable level sets using the Reeb graph. The major contributions of this chapter are as follows.

- We propose an interpolation scheme for point-based time-dependent density datasets with no connectivity information. The proposed interpolation scheme is capable of handling sparse data with large time intervals, while preserving the physical properties and the topology of the flow.
- We introduce a framework for extracting and analyzing the fluid interface topology. This framework is practical for the analysis of point-based multi-fluid datasets. It provides novel insights and tools for domain experts that facilitate further analysis and the estimation of solvent efficiency.

The remainder of the chapter is organized as follows. In section 6.1.1, we present related research on particle-based fluid simulation and topology-based feature tracking techniques. In section 6.2, suitable interpolation schemes are applied separately for time and spatial direction to develop a topologically clean extraction method for level sets. Section 6.3 presents examples and an analysis of our proposed visualization techniques. We include possible future work in the final section.

6.1.1 State of the art

A *level set* of a scalar field is given by the set of points with an identical scalar value. Our idea of studying the level sets of the density field was inspired by previous research on material boundary and fluid interface tracking, including the *front tracking (FT) method* (102), *level set method* (75), and *volume of fluid method* (39).

The FT method (38, 97, 102) advects the marked interface from an initial configuration and constantly retains the topology of the interface during the simulation. Therefore, this method is limited to topological changes in multi-phase-fluids, such as the merging or breaking of droplets.

However, the interface tracking algorithms mentioned above are not fully applicable to liquid-liquid extraction simulations because one of the fluid phases is fully dispersed and there is no continuous interface between the two fluids. Therefore, a material interface is not traceable in a slurry flow. Instead, topology-based techniques are more suitable for the analysis of level sets and more useful in this context because they can capture the absolute and relative behavior of small-scale features directly, such as dispersed bubbles (98).

The topology of 2D scalar fields can be studied using *contour trees* (27), which were introduced by de Berg and van Kreveld. Carr et al. (18) extended this approach with a simple and robust algorithm for the computation of contour trees. In three or more dimensions, a Reeb graph (83) is more efficient because it automatically encodes the behavior of level sets defined on an arbitrary manifold (98). Applications of the Reeb graph include fast isosurface extraction (19) and feature tracking (107).

Bremer et al. (16) proposed a hierarchical segmentation strategy for feature tracking in time-dependent isosurfaces. Isotherm extraction and vertex classification during the original times steps was carried out in discrete time slices based on structured spatial discretization. Unlike (16, 107), the extraction approach presented in the current study was directed at dynamic particle-based datasets, where the spatial structure changes randomly over time.

Particle-based fluid simulation is a particular class of numerical algorithms used to simulate fluids. Unlike traditional mesh-based simulation algorithms, particle-based methods are capable of handling complex flow problems where the geometry changes dramatically (30, 81). Smooth particle hydrodynamics (SPH) was the first particle method, which was introduced by Lucy (58) in 1977 to simulate astrophysical problems. Later, it was applied to the simulation of flow problems (68, 81) including free surfaces, multi-fluid interfaces, turbulent flow, and incompressible fluids, where grid-based methods were inadequate.

6.2 Reconstruction of the time-variable density field

Given a two-dimensional time-variable density field, we need to provide an appropriate interpolation scheme to obtain the corresponding density snapshots in sufficient time to perform accurate topology extraction and track phase movements. In order to use concepts from 3D visualization and convey an instantaneous representation of the unsteady field, we explored the use of a 2D time-dependent density field as a stationary 3D field where time was the third dimension. We examined a numerical dataset produced by a finite point set method (FPM) simulation of a two phase fluid, which contained seven time steps. The simulation technique advects the highly adaptive set of fluid particles that constitutes an implicit computational grid. However, particle creation and destruction means there is no correspondence between the particles from one time step to the next. Each of these computational particles carries information about the fluid density ρ and fluid velocity $\vec{v}$.

The reconstruction of a smooth time-dependent density field requires interpolation of the spatial and temporal axes. Our method is a hybrid of a moving least squares (MLS) approach for spatial interpolation and a Lagrangian-based scheme for temporal interpolation, which we derived from the FPM method. After considering the underlying equations of flow motion and applying Lagrangian advection to the computational particles, we obtained a physically meaningful interpolation of the scalar field over time. In the following subsections, we describe our scheme in detail.

6.2.1 Time interpolation

The given numerical dataset contains few time slices (7–10 available), so we propose the following algorithm to interpolate incrementally between two neighboring time steps T_0 and T_1 and obtain N intermediate steps, thereby increasing the resolution along the time-axis significantly. In general, our time interpolation consists of three steps. First, we re-initialize and re-sample a point cloud in the first intermediate time frame. Second, we update the density field in the new point cloud. Finally, we use the underlying physical model to re-compute the velocities of each re-sampled point. Given the flow fields $f, \vec{v}$ at time T_0 and T_1, we use an iterative technique to interpolate and re-compute $f, \vec{v}$ at intermediate time steps $T_0 + \Delta, ..., T_0 + N\Delta$, which is described in the following pseudocode.

Algorithm 3 Interpolation scheme

```
Input: $(x_i, \rho, \vec{v})_{T_0}$ and $(x_i, \rho, \vec{v})_{T_1}$
Output: $(x_i, \rho, \vec{v})_{T_0+j\Delta}$, for $0 < j \leq N$.
for $j = 1 \rightarrow N$ do
    $T^* \leftarrow T_0 + j\Delta$
    forward advect particles $x_i$ from $T_0 + (j-1)\Delta$ to $T^*$,
    backward advect particles $x_i$ from $T_1$ to $T^*$,
    recalculate velocities $(\vec{v}_i)_{T^*}$ at advected points using equation (6.2).
    resample particles at time $T$.
    for all $x_i$ do
        update density values $(\rho_i)_{T^*}$ and velocities $v_i^{T*}$ at resampled points using equation (6.1).
    end for
    $(x_i, \rho, \vec{v})_{T_{j\Delta}} \leftarrow (x_i, \rho, \vec{v})_{T^*}$
end for
```

In other words, for the first interpolation point $T^* = T_0 + \Delta$, we advect the particles from T_0 to T^* forward along the velocity field v_0 (see Figure 6.1) and we advect the particles backward from T_1 to T^*. The corresponding advection equations yield:

$$(x)^{T^*} = (x)^{T_0} + \vec{v^{T_0}}\Delta.$$

$$(x)^{T^*} = (x)^{T_1} - \vec{v^{T_1}}(T_1 - T_0 - \Delta).$$

Therefore, a new set of particles is formed. We resample these points into a regular grid and approximate the density function using Shepard's interpolation (94), as follows.

$$\rho(x) = \frac{\sum_{i=1}^{m} w_i \rho_i}{w_i} \tag{6.1}$$

Subsequently, we recompute the velocity vectors of the resampled points using a numerical estimation derived from the SPM method (100). In computational fluid dynamics, the flow properties are often described under two different reference frames, i.e., the Eulerian fixed reference frame and the Lagrangian co-moving reference frame. Unlike the Eulerian types of conservation laws, the Lagrangian specifications employ the idea of co-moving computational frames in the flow fields. Changes in the flow properties are described by the moving computational particles, which are known as *material derivatives*:

$$\frac{\mathrm{D}f}{\mathrm{D}t} = \frac{\partial f}{\partial t} + \vec{v} \cdot \nabla f$$

.

In the proposed algorithm, the fluid velocities are recomputed at the advected points. Therefore, we consider the momentum equation (100) in the material derivative form. Thus, a Lagrangian formulation of the momentum equation yields

$$\frac{D\vec{v}}{Dt} = -\frac{1}{\rho} + \frac{1}{\rho}\nabla \cdot (2\mu D) + \frac{1}{\rho}\vec{F}_s + \vec{g} \tag{6.2}$$

where μ is the fluid viscosity, which satisfies the mass conservation law. We use a continuous surface force (CSF) model (15) to define the surface tension force $\vec{F}_s = \sigma\kappa\vec{n}\delta_s$ with the interface normal vector $\vec{n}$, curvature κ, and normalized surface delta function δ_s. The values of $\vec{n}, \kappa, and \delta_s$ can be obtained by introducing marking functions for fluid types 1 and 2. For details, please refer to (15, 100). The velocities at the resampled points during the intermediate time step T^* are computed component-wise by solving the following equation using Chorin's projection method (24).

$$\vec{v} - \frac{dt}{\rho}\nabla\mu \cdot \nabla\vec{v} - dt\frac{\mu}{\rho}\Delta\vec{v} = \vec{v^n} + dt(\frac{F_S^n}{\rho} + g + \frac{\nabla\mu}{\rho} \cdot \nabla(\vec{v^n})) \tag{6.3}$$

After resampling the points at T^* and re-evaluating the velocity field, we then use T^* as the new T^0 and repeat the computations to interpolate the flow fields at $T_0 + 2\Delta$.

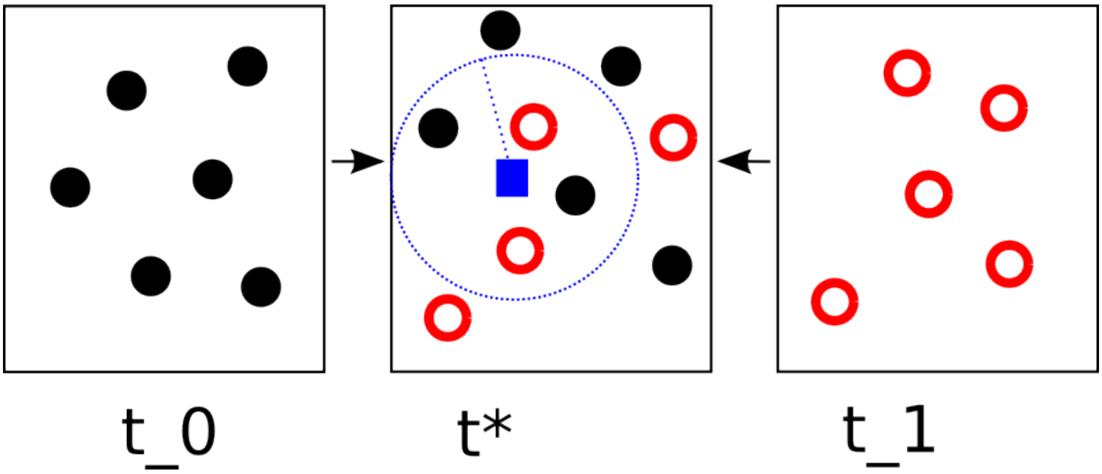


Figure 6.1: Proposed interpolation scheme: forward advection of fluid particles from the previous time step to intermediate stage t^* and recalculation of the velocity field. Backward advection of fluid particles from the last time step and recalculation of the velocity field. The data are resampled at the intermediate time t^* at the blue points using a radial basis function.

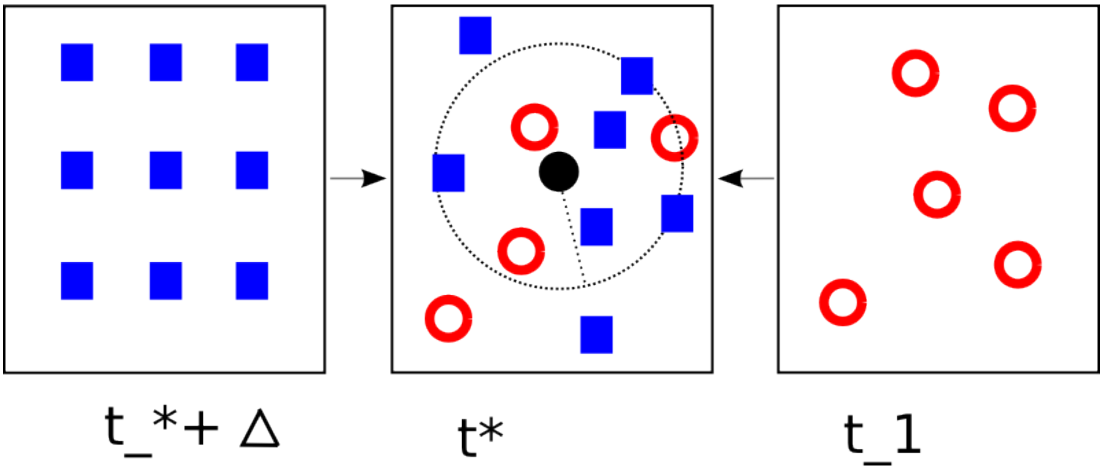


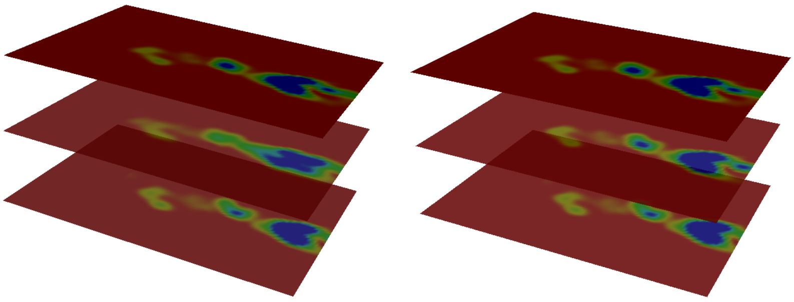
Figure 6.2: Left: the first inserted time frame where the values were resampled at regular points. The second inserted frame was created by advecting particles from $T_0 + \Delta$ and T_1. We the resampled the points in the same grid used at $T_0 + \Delta$.

Two main characteristics distinguish our method from current interpolation schemes: recalculation of the velocity field of the resampled points using a physical model and iterative advection of the resampled points to the next interpolated time frame using the recomputed velocity. While resampling the points and interpolating the density fields, it is important to reconstruct the velocity field at intermediate time frames appropriately. To obtain a physically meaningful interpolation, we use the velocity updating schemes proposed for the SPM method. By solving the original momentum equation (6.2) in the simulation model, we obtain a good interpolation by reconstructing the missing time steps.

The additional computational effort required to recalculate the velocities is justified by the quality of the results, as demonstrated in the following example. We compared our method with a linear interpolation scheme without velocity recomputation. A naive way of interpolating the density field between two time steps is a linear combination of the two, such as:

$$f(t_0 + \Delta t) = \frac{t_0 - t_1 + \Delta t}{t_0 - t_1} f(t_0) + \frac{-\Delta t}{t_0 - t_1} f(t_1).$$

. However, linear interpolation yields physically incorrect results because it fails to consider the flow, which leads to errors during tracking or topology extraction. As shown in Figure 6.3a, the middle layer is a linear interpolation of the two slices. In our proposed velocity recalculation method, the interpolation provides a physical, meaningful description of the fluid properties from one time step to the next (6.3b).



(a) Left: the slice in the middle is a linear blending of the top and bottom slices.

(b) Right: the slice in the middle was interpolated using our proposed method.

Figure 6.3: Linear and nonlinear interpolation

6.2.2 Spatial Approximation

At each given time step T_i, the positions of the point clouds and the number of points are variable. In order to achieve a continuous representation of the field that can be resampled by a uniform distribution of spatial points, we perform a MLS approximation to fit the velocity and density values.

The MLS method is one of the most popular mesh-free interpolation schemes and it was introduced by Lancaster and Salkauskas (51) in 1981 to derive a smooth surface from a set of scattered data points in space. To interpolate flow quantities at a given evaluation point, we construct a weighted least squares fit based on a consideration of a local set of neighboring points. A global approximation is obtained by moving this evaluation point across the whole domain. The differentiability of the global fitting function is ensured by selecting a continuously differentiable weight function (also known as a kernel function), see Levin (54). For a finite set of points $\mathbf{x}_i$, we approximate the given scalar value f_i using a polynomial $f(\mathbf{x})$ at $\mathbf{x} = (x, y, z)$ to minimize the weighted square error:

$$G(x) = \min \sum_i w(r_i) \| f(\mathbf{x}_i) - f_i \|^2 \tag{6.4}$$

where f is a n degree polynomial in m-dimensional space and the weight w is a scaling function that depends on the distance $r_i = \|\mathbf{x} - \mathbf{x}_i\|$. For example, a quadratic approximation of 2D scattered data can be written as follows.

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \cdot \mathbf{c} = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix}$$

To address the problem of variable point densities during different time steps, the spatial interpolation can be controlled by adjusting the size of the neighborhood radius r. We will specify the selection of this parameter later. First, let us consider how the minimization problem can be solved.

The minimization problem (6.4) can be solved by setting the partial derivatives with respect to c_i as zero, $\dfrac{\partial G}{\partial c_i} = 0$.

As a result, the coefficients c_i can be obtained by solving the following linear system,

$$\sum_i w(r_i) \mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T \mathbf{c} = \sum_i w(r_i) \mathbf{b}(\mathbf{x}_i) f_i, \tag{6.5}$$

and the solution of the coefficient c_i yields

$$\mathbf{c} = [\sum_i w(r_i) \mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T]^{-1} \sum_i w(r_i) \mathbf{b}(\mathbf{x}_i) f_i. \tag{6.6}$$

The idea of using a weight function is based on the fact that points closer to point $\mathbf{x}$ should have greater effects than distant points, as shown in Figure 6.4.

Therefore, the intuitive choices for weight functions are radial basis functions that depend on the inverse distance. Cubic splines and Gaussian functions are commonly used in mesh-free/particle-based numerical simulations such as smoothed particle hydrodynamics (13), see (30, 49, 100).

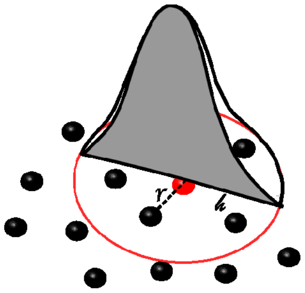


Figure 6.4: Neighborhood structure of a given point inside the point cloud.

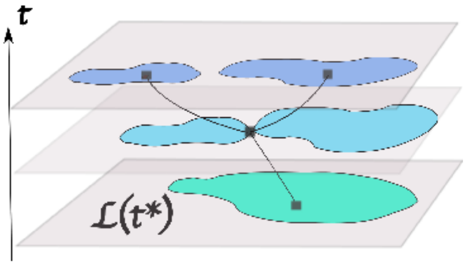


Figure 6.5: Level sets of a function and its Reeb graph.

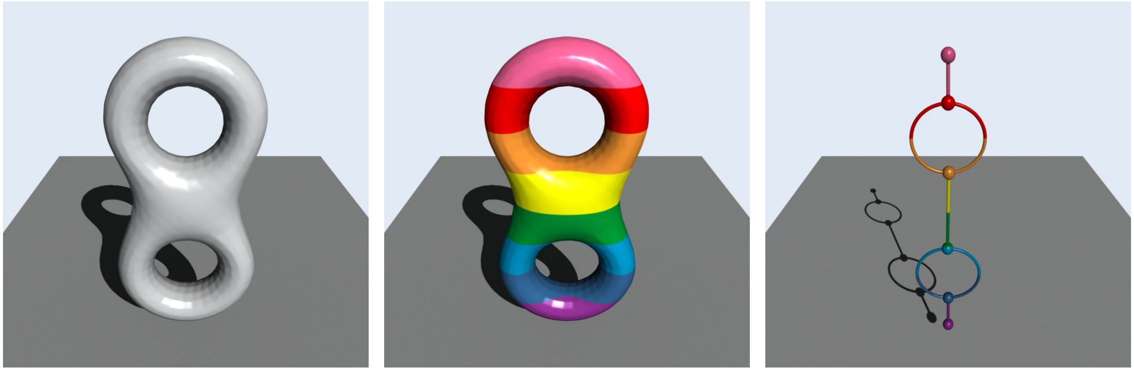
To define a good weight function, we need to restrict the number of points by imposing a small compact support on the weight function, so points outside the compact support will have zero influence. Moreover, a minimal number of points should be guaranteed in order to produce a nonsingular system of equations for a second order approximation (30).

To ensure consistency with the FPM simulation data, we select a Gaussian weight function. The size of h is set so there are at least six points in a circle with the radius h:

$$w(\mathbf{x}-\mathbf{x}_i,h)=\begin{cases}\exp(-2\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h^2}), & \text{if } \frac{\|\mathbf{x}-\mathbf{x}_i\|}{h}\leq 1\\ 0, & \text{elsewhere.}\end{cases}$$

To approximate vector fields of the flow, e.g., velocity, we apply the MLS method to each component of the vector. In our implementation, we used a quadratic basis function for $b(\mathbf{x}_i)$. If there are not enough points in the neighborhood, the basis function is reduced to a linear function.

Previously, we obtained a continuous density field in spatial directions using MLS and in the time direction using a Lagrangian method. In the next section, we discuss the use of Reeb graphs for analyzing this interpolated density field.



(a) Torus Manifold.

(b) Level sets of the height field. Each ribbon color indicates one level set in the height field

(c) Reeb graph of the height function showing the dynamics of the level sets

Figure 6.6: A double torus example of a Reeb graph

6.3 Reeb Graph

After constructing a continuous density function over time, we can extract a spatiotemporal isosurface in all three dimensions for any given density value. We perform Reeb graph extraction to analyze the topology changes in these surfaces using time as a Morse function.

We recall the definition of a Reeb graph (see (83, 98) for more details), as follows. A real-valued function $f : M \to \mathcal{R}$ is defined on a manifold M. The Reeb graph is a graph that describes the topology change in the level sets of f. For a given value t^*, the level set $L(t^*)$ (Figure 6.5) is defined as the inverse image of t^* on M through f:

$$L(t^*) = f^{-1}(t^*)$$

.

Each connected component of the level set $L(t^*)$ is known as a contour, see Figure 6.5. In Morse theory (66), topological changes in contours only occur at the critical points of f (79). A Reeb graph has nodes that correspond to the critical points of f

while the edges represent the connectivity of the contours of f, and t^* evolves continuously in $\mathcal{R}$, see Figure 6.5.

If we use a double torus as an example (see Figure 6.6a) and let the height field z be our scalar function, the connected level sets $L(z)$ of z are contours with a certain height value $z = z^*$(shown in 6.6b). There are four types of nodes on a Reeb graph, which correspond to four different events during contour evolution.

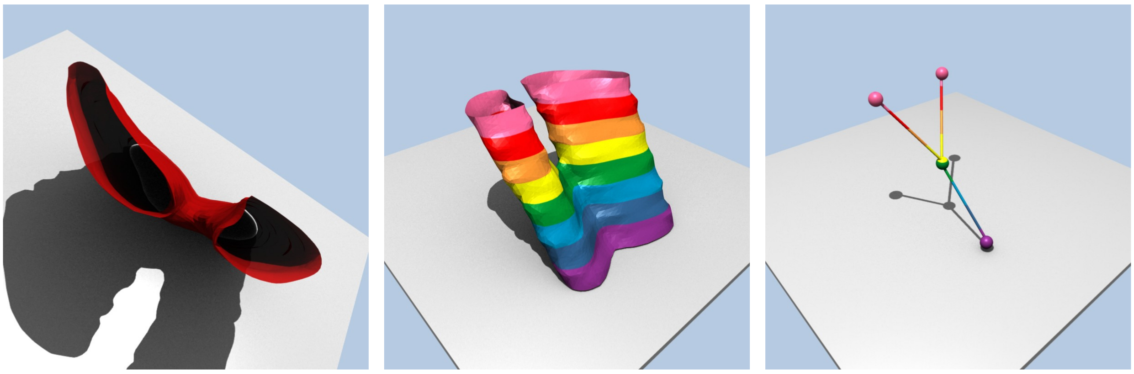
Birth node A birth node in a Reeb graph corresponds to a minimum critical point of f. In the torus example, this is the point where the bottom purple node is shown in Figure 6.6c. A contour of the scalar function z emerges for the first time from the position of the birth node, hence its name.

Bifurcation node The blue node on the torus example is a bifurcation node. As the name implies, it is a node where the contour of a given scalar function splits. In mathematical terms, a bifurcation node is a type of saddle point where its second derivative degenerates.

Merging node In contrast to the bifurcation case, the situation where two contours merge into one is characterized as a merging node. (See the second node from the top in Figure 6.6c). It may be viewed as symmetric to the bifurcation node with respect to time reversal (inverting the z direction). Therefore, merging and bifurcation nodes are topologically equivalent and both correspond to a saddle point on the manifold $\mathcal{M}$.

Death node The pink node at the very top of Figure 6.6c is a death node on the torus. At this point, the contours of z vanish, so it is known as a death node. A death node corresponds to a maximum critical point of the scalar function z. Similar to the bifurcation/merging pair, death nodes can be considered symmetric to birth nodes.

Extraction of the high density area and computation of the Reeb graph We are interested in regions where the density of the solvent is above a certain threshold ε (well-mixed regions with high concentrations of the solvent phase), so we extract an implicit surface $f(\mathbf{x}) = \varepsilon$ that encloses regions with a higher solvent density. Each segment of the surface (which is perpendicular to the time axis) corresponds to the



(a) Bifurcation: regions splitting after a certain time point. The outer normals of the plane points are in the time direction.

(b) Level sets of time.

(c) Extracted Reeb graph for the bifurcation case.

Figure 6.7: Bifurcation example.

original given 2D slices or the interpolated intermediate slices. Therefore, the level sets of the implicit surface yield isocontours on the original 2D data. As a straightforward interpretation, our surface can be interpreted as a sweeping of the 2D isocontours.

Using a standard Marching Cubes algorithm (57), we extracted the time surface from the interpolated density function discussed in the previous chapter. The results are shown in Figure 6.7, 6.8 and 6.9. A tracking Reeb graph was computed for this time surface using time as the Morse function (similar to the height field in the torus example). We used a vtkReebGraph Filter based on Pascucci's streaming algorithm (77, 78). Note that the Reeb graph algorithm operates on a closed surface, so we introduced ghost cells in the first and last time slices, as well as the ingoing and outgoing flow boundaries.

6.4 Results

Figures 6.7a, 6.8a, and 6.9a show the extracted isosurfaces (swept level sets over time). The corresponding Reeb graphs are shown in Figures 6.7c, 6.8c, and 6.9c. Figures 6.7b, 6.8b, and 6.9b are the colored level sets over time, which provide an intuitive understanding of how the flow propagates over time. The node coordinates (x, y, t) indicate that one of the four types of contour evolution events occurs at point (x, y) for time t.

Moreover, the edge connectivity of the Reeb graph defines the type of nodes and indicates the connectivity of contours. We discuss the results using the following three examples. The first example (Figure 6.7) is a typical case where the high density flow regions split into two. As shown in Figure 6.7c, a bifurcation event occurs between the yellow and green level sets, which points to the contour splitting event in Figure 6.7b. In contrast to bifurcation events, another example (Figure 6.8) shows a case where nodes meet and merge to produce a larger one.

During the classification of Reeb graph nodes, the birth and death nodes both have multiple meanings that depend on the positions of the nodes, i.e., those on the ingoing boundary of the flow show that a new contour is entering the domain because of the feed-in fluid. Similarly, for death nodes, those that are on the outgoing boundary show that the flow is exiting the area whereas those inside the *xy*-plane indicate an event where fluid blobs smear out and vanish.

A complete scenario is shown in Figure 6.9. The white plane at the bottom corresponds to the *xy*-plane. The outer normal points up and indicates the time direction. The flow is coming in from the side of the red plane and it exits freely at the left side. This example was generated from ten time slices with one intermediate interpolation step between each of the slices. All four node types occur in this scenario, e.g., the circled orange node is a birth node that indicates an increase in the density of the second fluid phase in the region. Compared with the direct coloring of level sets (Figure 6.9b), the Reeb graph on the right side (Figure 6.9c) provides a better illustration of the topological skeleton of the surface. Moreover, we also show a comparison between the conventional graph layout (Figure 6.10) and our 3D Reeb graph representation (Figure 6.9c). Both representations contain exactly the same information, but the 3D representation is clearly better at illustrating the topological evolution.

Furthermore, the visualization technique presented in this paper facilitates visual tracking of fluid bubbles in a stationary view. It also facilitates further analysis and an estimation of the solvent efficiency, such as finding the regions where no solvent mixing occurred.

6.5 Conclusions and future research possibilities

In this chapter, we presented a method for characterizing the evolution of time-dependent density fields using Reeb graphs. The proposed method can be utilized by mechanical engineers to evaluate the design of mixing devices and to adjust the choice of inflow velocities to optimize the mixing results. The challenge of our dataset was the low time resolution, which can lead to a low tracking accuracy and an ill-defined topology. To improve the quality of our topology tracking, therefore, we proposed the application of Lagrangian advection to the point set where the extra velocity information was used to obtain a physically-based interpolation of the density field.

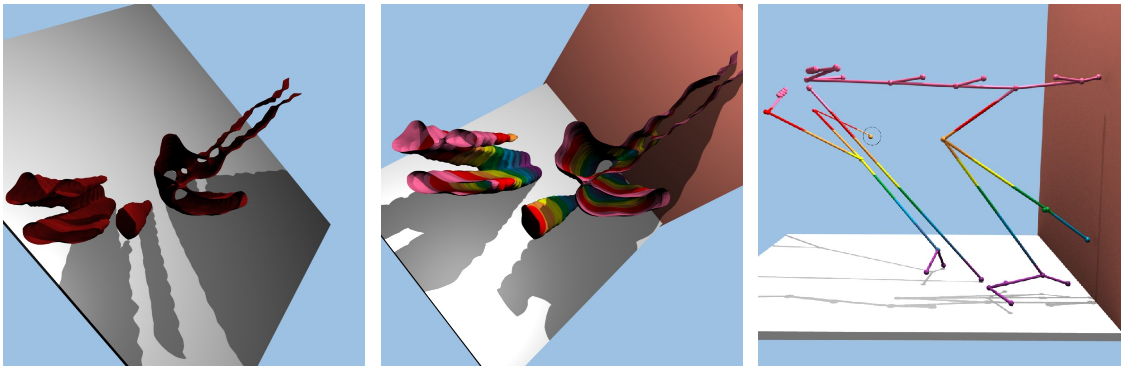
Possible future work includes extending our approach to a 3D simulation of multi-fluid data, which will require the computation of a Reeb graph for a scalar function t over a 4D manifold. Techniques such as tracking the Jacobi set of a time-dependent scalar field could also be a good choice for representing topology changes in the density field. Further work could also be done to evaluate how the time resolution of the input data affects the extracted topology. Apparently, the topology of the extracted surface will change if the time steps are large. Thus, finding a threshold for the time stepping length (below which the extracted Reeb graph does not change) will probably be related to the maximum gradient of the velocity field and the Courant-Friedrichs-Lewy (CFL) conditions of the numerical scheme.

(a) Merging: regions merge to produce a large region after a certain time point.

(b) Level sets of time.

(c) Extracted Reeb graph for the merged case.

Figure 6.8: Example of merged regions.



(a) Extracted surface for the whole domain for all time steps.

(b) Level sets of time. The colored wall is the incoming boundary of the flow.

(c) Extracted Reeb graph of the left image.

Figure 6.9: A complete scenario. The white plane denotes the spatial domain. The flow is entering the domain from the red wall.

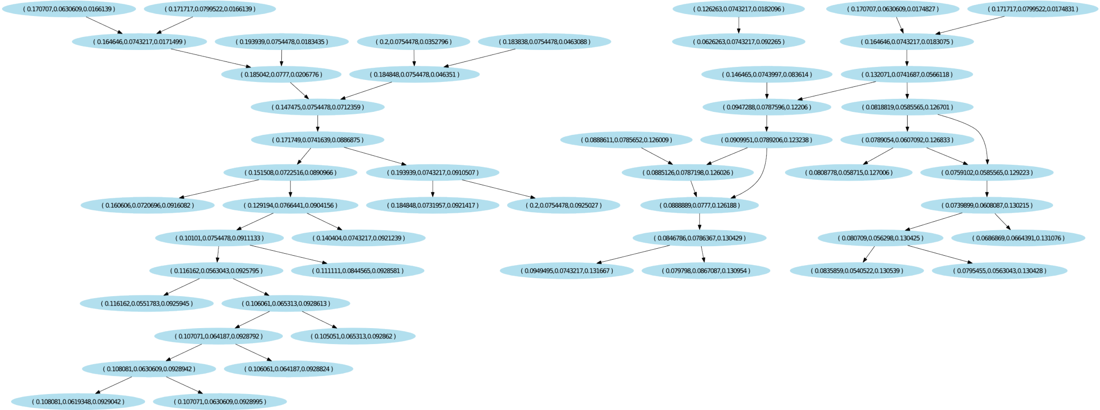


Figure 6.10: Conventional graph layout. The spatial and time coordinates are printed inside the nodes. This representation is convoluted and hard to understand. By contrast, our 3D Reeb graph representation provides an intuitive understanding of how the topology changes.

Chapter 7

Conclusions and future work

In this study, we simulated a two phase fluid in an extraction column using CFD and PBM models. We developed stochastic and feature-based visualization techniques for visualizing the distribution of droplets and their dynamic behavior. The fluid interface was our main focus because it is a major issue during multi-phase fluid visualization. We developed a novel visualization technique for analyzing complex fluid interface behaviors over time. Further investigations of the fluid interface were conducted using a topology-based method. We proposed a physically plausible scheme for interpolating fluid volume fraction data using a randomly distributed point cloud.

In future studies, we plan to take the topological analysis of the fluid interface to higher dimensions. The application of Reeb graphs to multi-dimensional data may provide new insights into droplet behavior. Uncertainty visualization could also be a new direction for addressing droplet dynamics, especially during breakage and collision modeling.

References

[1] OPEN FOAM (open field operation and manipulation). http://www.openfoam.com, September 2011. Open CFD Ltd./SGI. 67

[2] P. Albrecht. The runge-kutta theory in a nutshell. *SIAM J. Numer. Anal.*, 33:1712–1725, 1996. 34

[3] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy. Smooth, volume-accurate material interface reconstruction. *IEEE Trans. Vis. Comput. Graphics*, 16(5):802 –814, 2010. 41, 52, 53, 56

[4] R. Andersson and B. Andersson. Modeling the breakup of fluid particles in turbulent flows. *AIChE Journal*, 52(6):2031–2038, 2006. 16, 19

[5] E. Aulisa, S. Manservisi, and R. Scardovelli. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *Journal of Computational Physics*, 188(2):611–639, 2003. 15

[6] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM computing surveys*, 23(3):345–405, 1991. 34

[7] A. W. Bargteil, T. G. Goktekin, J. F. O'Brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graphic*, 25(1):19–38, 2006. 53

[8] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien. A texture synthesis method for liquid animations. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 345–351, 2006. 53

[9] H.-J. Bart. Reactive extraction in electrical fields. *Finely Dispersed Particles: Micro-, Nano- and Atto-Engineering*, pages 465–491, 2005. 18

[10] H.-J. Bart, M. Hlawitschka, M. Mickler, M. Jaradat, S. Didas, F. Chen, and H. Hagen. Tropfenschwarmanalytik, visualisierung und simulation. *Chemie Ingenieur Technik*, 83(7):965– 978, 2011. 25

[11] G. K. Batchelor. *An introduction to fluid dynamics.* Cambridge University Press, Cambridge Mathematical Library series, 2000. 15

[12] C. Bennis, J.-M. Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. *SIGGRAPH Comput. Graph.*, 25:237–246, July 1991. 60

[13] J. Bonet and S. Kulasegaram. Alternative total lagrangian formulations for corrected smooth particle hydrodynamics (csph) methods in large strain dynamics problems. *European Review of Finite Elements*, 11:893–912, 2002. 97

[14] K. S. Bonnell, M. A. Duchaineau, D. R. Schikore, B. Hamann, and K. I. Joy. Material interface reconstruction. *IEEE Trans. Vis. Comput. Graphics*, 9:500–511, 2003. 41, 52, 53, 56, 68

[15] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100:335–354, June 1992. 43, 93

[16] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16:248–260, March 2010. 90

[17] K. Buerger, F. Ferstl, H. Theisel, and R. Westermann. Interactive streak surface visualization on the GPU. *IEEE Trans. Vis. Comput. Graphics*, 15:1259–1266, 2009. 53, 65

[18] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 918–926, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics. 90

[19] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 497–504, Washington, DC, USA, 2004. IEEE Computer Society. 12, 90

[20] J. R. Cash and A. H. Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Trans. Math. Softw.*, 16:201–222, 1990. 61

[21] F. Chen and H. Hagen. A Survey of Interface Tracking Methods in Multi-phase Fluid Visualization. In A. Middel, I. Scheler, and H. Hagen, editors, *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, volume 19 of *OpenAccess Series in Informatics (OASIcs)*, pages 11–19, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 39

[22] F. Chen, M. Hlawitschka, H.-J. Bart, and H. Hagen. Innovative multiphase fluid visualization for droplet column simulations. *Proceedings of " first international symposium on Multi-scale Multiphase Process Engineering (MMPE)*, October 2011. 25

[23] F. Chen, H. Obermaier, H. Hagen, B. Hamann, J. Tierny, and V. Pascucci. Topology analysis of time-dependent multi-fluid data using the reeb graph. *Computer Aided Geometric Design*, 2012. 87

[24] A. J. Chorin. Numerical solution of the Navier-Stokes Equations. *Math. Comp.*, 22:745–762, 1968. 93

[25] C. Coulaloglou and L. Tavlarides. Description of interaction processes in agitated liquid-liquid dispersions. *Chemical Engineering Science*, 32(11):1289 – 1297, 1977. 16, 19

[26] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on Pure and Applied Mathematics*, 5(3):243–255, 1952. 47

[27] M. de Berg and M. J. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18(3):306–323, 1997. 90

[28] C. Drumm, M. Attarakih, M. W. Hlawitschka, and H.-J. Bart. One-group reduced population balance model for CFD simulation of a pilot-plant extraction column. *Industrial & Engineering Chemistry Research*, 49(7):3442–3451, 2010. 16, 19

[29] C. Drumm and H.-J. Bart. Hydrodynamics in a rdc extractor: Single and two-phase piv measurements and CFD simulations. *Chemical Engineering & Technology*, 29(11):1297–1302, 2006. 16, 17

[30] C. Drumm, S. Tiwari, J. Kuhnert, and H.-J. Bart. Finite pointset method for simulation of the liquid liquid flow field in an extractor. *Computers & Chemical Engineering*, 32(12):2946 – 2957, 2008. 88, 90, 97, 98

[31] X. Duan, Y. Ma, and R. Zhang. Optimal shape control of fluid flow using variational level set method. *Physics Letters A*, 372(9):1374 – 1379, 2008. 44

[32] D. Enright, R. P. Fedkiw, and F. J. H. et al. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183:83–116, 2002. 41, 44, 45

[33] R. V. Garimella and V. e. a. Dyadechko. *Interface Reconstruction in Multi-fluid, Multi-phase Flow Simulations*, pages 19–32. Springer, 2005. 41

[34] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. Joy. Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Trans. Vis. Comput. Graphics*, 14(6):1404 –1411, 2008. 26, 28, 53

[35] J. Glimm, J. W. Grove, X. L. Li, and N. Zhao. Simple front tracking, 1999. 40

[36] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*. Springer-Verlag, New York, NY, USA, 1996. 47

[37] M. Goodson and M. Kraft. Simulation of coalescence and breakage: an assessment of two stochastic methods suitable for simulating liquid-liquid extraction. *Chemical Engineering Science*, 59(18):3665 –3881, 2004. 25

[38] O. Hänel, L. Tran, and O. Vilsmeier. A front tracking method on unstructured grids. *Computers & Fluids*, 32(4):547 – 570, 2003. 40, 90

[39] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39(1):201–225, 1981. 40, 41, 45, 47, 55, 90

[40] M. Hlawitschka, M. Attarakih, F. Chen, M. Jaradat, M. Mickler, C. Drumm, and H.-J. Bart. Simulation of extraction columns of type kühni with dpbm. In *Proceedings of 4th International Conference on Population Balance Modelling (PBM 2010)*, September 2010. 15

[41] M. Hlawitschka, F. Chen, H.-J. Bart, and H. Hagen. CFD simulation und verbesserte datenauswertung einer extraktionskolonne vom typ kühni. *Proc. of first Young Researcher Symposium (YRS)*, 2011. 15, 25

[42] M. W. Hlawitschka, M. Mickler, and H.-J. Bart. Simulation einer gerührten miniplant-extraktionskolonne mit hilfe eines gekoppelten CFD-populationsbilanzmodells. *Chemie Ingenieur Technik*, 82(9):1389–1390, 2010. 16, 17

[43] D. W. Holmes, J. R. Williams, and P. Tilke. Smooth particle hydrodynamics for grain scale multi-phase fluid simulations. *International Conference on Particle-Based Method*, pages 237–240, 2009. 15

[44] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proc. of IEEE Vis '92*, pages 171–178, 1992. 52

[45] M. Ikits, J. Kniss, A. E. Lefohn, and C. Hansen. Volume rendering techniques. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, pages 667–692. Addison-Wesley, 2004. 66

[46] J. E. P. Jr. and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465 – 502, 2004. 45, 46

[47] S. A. King, R. A. Crawfis, and W. Reid. Fast volume rendering and animation of amorphous phenomena. In M. Chen, A. E. Kaufman, and R. Yagel, editors, *Volume Graphics*. Springer, London, 2000. 66

[48] H. Krishnan, C. Garth, and K. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *IEEE Trans. Vis. Comput. Graphics*, 15:1267–1274, 2009. 53, 61

[49] J. Kuhnert and S. Tiwari. *Finite pointset method based on the projection method for simulations of the incompressible Navier-Stokes equations*, volume 26, chapter 1, pages 373–388. Springer, Berlin, ix edition, 2003. 15, 32, 97

[50] D. Lakehal, M. Meier, and M. Fulgosi. Interface tracking towards the direct simulation of heat and mass transfer in multiphase flows. *Int. J. Heat Fluid Flow*, 23:242–257, 2002. 41, 42, 44, 55

[51] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981. 96

[52] R. Laramee, C. Garth, J. Schneider, and H. Hauser. Texture advection on stream surfaces: a novel hybrid visualization applied to CFD simulation results. *Proc. of Eurographics 06'*, pages 155–162, 2006. 53

[53] R. S. Laramee, J. Schneider, and H. Hauser. Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics. In *Proc. of Eurographics 04'*, pages 85–90, 2004. 53

[54] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67:1517–1531, 1998. 97

[55] H. Levy and F. Lessman. *Finite Difference Equations*. Dover, New York, 1992. 47

[56] A. Lez, A. Zajic, K. Matkovic, A. Pobitzer, M. Mayer, and H. Hauser. Interactive exploration and analysis of pathlines in flow data. In *Proc. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2011)*, pages 17–24, 2011. 26, 28

[57] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21:163–169, August 1987. 101

[58] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, December 1977. 90

[59] H. Luo and H. F. Svendsen. Theoretical model for drop and bubble breakup in turbulent dispersions. *A.I.Ch.E Journal*, 42(5):1225–1233, 1996. 11, 20

[60] D. L. Marchisio, J. T. Pikturna, R. O. Fox, R. D. Vigil, and A. A. Barresi. Quadrature method of moments for population balance equation describing the hydrodynamics of interacting liquid-liquid dispersions. *AL. Che. J.*, 49:1266–1276, 2003. 19

[61] M. Marek, W. Aniszewski, and A. Boguslawski. Simplified volume of fluid method for two-phase flows. *Task Quarterly*, 3:255 – 265, 2008. 41

[62] L. Margolin, J. M. Reisner, and P. K. Smolarkiewicz. Application of the volume-of-fluid method to the advection and condensation problem. *American Meteorological Society*, 125:2265–2273, 1997. 45

[63] C. Martinez-Bazán, J. L. Monatenes, and J. C. Lasheras. On the breakup of an air bubble injected into a fully developed turbulent flow. part 1. breakup frequency. *Journal of Fluid Mechanics*, 401:147 – 182, 1999. 16

[64] P. Meakin, A. M. Tartakovsky, T. Scheibe, D. M. Tartakovsky, G. Redden, P. E. Long, S. C. Brooks, and Z. Xu. Particle methods for simulation of subsurface multiphase fluid flow and biogeochemical processes. In *J. Phys.: Conf. Ser. 78, 012047*, SciDAC 2007: Boston, Massachusetts, USA, Jun 2007. 15

[65] J. S. Meredith and H. Childs. Visualization and analysis-oriented reconstruction of material interfaces. *Comput. Graph. Forum*, 29(3):1241–1250, 2010. 52

[66] J. Milnor. Morse theory. *Annals of mathematical studies*, 51:847–849, 1963. 99

[67] G. Modes and H. J. Bart. CFD-simulation der strömungsnichtidealitäten der dispersen phase bei der extraktion in gerührten extraktionskolonnen. *Chemie Ingenieur Technik*, 73(4):332–338, 2001. 16

[68] J. J. Monaghan. Simulating free surface flows with sph. *J. Comput. Phys.*, 110:399–406, February 1994. 48, 90

[69] M. Müller. Fast and robust tracking of fluid surfaces. In *Proc. of '09 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 237–245, New York, NY, USA, 2009. ACM. 41

[70] F. Neyret. Advected textures. In *Proc. of the '03 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–153, 2003. 53

[71] W. Noh and P. Woodward. Simple line interface method. In *Proc. of 5th international conference on fluid dynamics*, volume 59, pages 330–340, 1976. 41, 52, 53, 56

[72] H. Obermaier, F. Chen, H. Hagen, and K. I. Joy. Visualization of material interface stability. In *IEEE Pacific Visualization (PacificVis 2012)*, pages 225–232, 2012. 51

[73] S. Osher. A level set formulation for the solution of the dirichlet problem for hamilton-jacobi equations. *SIAM J. Math. Anal.*, 24:1145–1152, September 1993. 43

[74] S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys*, 169(2):463–502, May 2001. 40, 41, 42, 44, 47

[75] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988. 40, 44, 90

[76] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit surfaces*. Springer, 2003. 41, 44, 47

[77] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(1):249–268, 2003. 101

[78] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 58, New York, NY, USA, 2007. ACM. 101

[79] G. Patan, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the reeb graph. *IEEE Trans. Vis. Comput. Graph.*, 15(4):583–595, 2009. 99

[80] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A pde-based fast local level set method. *Journal of Computational Physics*, 155(2):410 – 438, 1999. 44

[81] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-Based Simulation of Fluids. *Computer Graphics Forum*, 22(3):401–410, 2003. 48, 90

[82] M. J. Prince and H. W. Blanch. Bubble coalescence and break-up in air-sparged bubble columns. *A.I.Ch.E Journal,*, 36(10):1485 –1499, 1990. 16

[83] G. Reeb. Sur les points singuliers d'une forme de pfaff complétement intégrable ou d'une fonction numérique. *Computes-rendus de l'Académie des Sciences*, 222:847–849, 1946. 90, 99

[84] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *J. Comput. Phys.*, 141:112–152, April 1998. 41, 52, 56

[85] R. Rieger, C. Weiss, G. Wigley, H.-J. Bart, and R. Marr. Investigating the process of liquid-liquid extraction by means of computational fluid dynamics. *Computers & Chemical Engineering*, 20(12):1467 – 1475, 1996. 16

[86] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer. Pathline predicates and unsteady flow structures. *The Visual Computer*, 24:1039–1051, 2008. 26

[87] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based stream surfaces and path surfaces. In *Proc. of Graphics Interface '07*, pages 289–296, 2007. 52

[88] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy, and W. Kollmann. A tetrahedra-based stream surface algorithm. In *Proc. of IEEE VIS '01*, pages 83–91, 2001. 52

[89] N. Z. Schiller L. A drag coefficient correlation. *Z. Ver. Deutsch. Ing.*, 77:318–320, 1935. 17

[90] J. A. Sethian. Curvature and the evolution of fronts. *Commun. Math. Phys.*, 101:487–499, 1985. 41

[91] J. A. Sethian and P. Smereka. Level set methods for fluid interfaces. *Annu. Rev. Fluid Mech.*, 35:341–372, 2003. 41

[92] A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2001. 60, 63

[93] A. Sheffer and J. C. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *Proc. of IEEE VIS '02*, pages 291–298, 2002. 60

[94] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM. 93

[95] S. Shin. Computation of the curvature field in numerical simulation of multiphase flow. *J. Comput. Phys.*, 222:872–878, March 2007. 43

[96] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *J. Comput. Phys*, 162:301–337, 2000. 15, 41, 45

[97] H. Terashima and G. Tryggvason. A front-tracking/ghost-fluid method for fluid interfaces in compressible flows. *J. Comput. Phys.*, 228:4012–4037, June 2009. 40, 90

[98] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1177–1184, 2009. 90, 99

[99] S. Tiwari and J. Kuhnert. A meshfree method for incompressible fluid flows with incorporated surface tension. *Revue europenne des lments finis*, 11:965–087, 2002. 43

[100] S. Tiwari and J. Kuhnert. Modeling of two-phase flows with surface tension by finite pointset method (fpm). *J. Comput. Appl. Math.*, 203:376–386, June 2007. 43, 48, 93, 97

[101] G. Turk. Texture synthesis on surfaces. In *Proc. of SIGGRAPH '01*, pages 347–354, 2001. 53, 60

[102] S. O. Unverdi and G. Tryggvason. A front tracking method for viscous incompressible multi-fluid flows. *J. Comput. Phys*, 100:25–37, 1992. 40, 55, 90

[103] J. J. van Wijk. Implicit stream surfaces. In *Proc. of VIS '93*, pages 245–252, 1993. 52

[104] J. J. Van Wijk. Image based flow visualization for curved surfaces. In *Proc. of IEEE VIS '03*, pages 123–130, 2003. 53

[105] A. Vikhansky and M. Kraft. Modelling of a RDC using a combined CFD-population balance approach. *Chemical Engineering Science*, 59(13):2597 – 2606, 2004. 16

[106] M. W. Hlawitschka, M. Jaradat, F. Chen, M. Attarakih, J. Kuhnert, M. Mickler, and H. J. Bart. A CFD-population balance model for the simulation of kühni extraction column. *Computer Aided Chemical Engineering*, pages 66–70, 2011. 15

[107] G. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci. Feature tracking using reeb graphs. In *DOE Computer Graphics Forum Montery, CA*, 2009. 12, 90

[108] D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. In *Numerical methods for fluid dynamics*, pages 273–285. Academic Press, 1982. 41, 52