Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes

Marek K. Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund,
Brian Bunch Christensen, J. Andreas Bærentzen, and Robert Bridson

Abstract—In this paper, we present a method for animating multiphase flow of immiscible fluids using unstructured moving meshes. Our underlying discretization is an unstructured tetrahedral mesh, the deformable simplicial complex (DSC), that moves with the flow in a Lagrangian manner. Mesh optimization operations improve element quality and avoid element inversion. In the context of multiphase flow, we guarantee that every element is occupied by a single fluid and, consequently, the interface between fluids is represented by a set of faces in the simplicial complex. This approach ensures that the underlying discretization matches the physics and avoids the additional book-keeping required in grid-based methods where multiple fluids may occupy the same cell. Our Lagrangian approach naturally leads us to adopt a finite element approach to simulation, in contrast to the finite volume approaches adopted by a majority of fluid simulation techniques that use tetrahedral meshes. We characterize fluid simulation as an optimization problem allowing for full coupling of the pressure and velocity fields and the incorporation of a second-order surface energy. We introduce a preconditioner based on the diagonal Schur complement and solve our optimization on the GPU. We provide the results of parameter studies as well as a performance analysis of our method, together with suggestions for performance optimization.

Index Terms—

1 Introduction

In this paper, we present a finite element method for animating multiphase flow of immiscible liquids (i.e. non-mixing liquids, such as water and oil, see Figure 1). Our approach is based on the Lagrangian deformable simplicial complex (DSC) method, previously used for topology-adaptive deformable interface tracking [1], [2], which trades the apparent simplicity of the level set method for robustness and support of multiple phases, as well as offering an unstructured, moving computational grid. Each element in the mesh is assigned a single material and interfaces are composed of faces in the DSC. The DSC moves with the fluid in a Lagrangian fashion and a variety of mesh optimization operations improve element quality and avoid element inversion. We formulate the solution of the Navier-Stokes equations in terms of a quadratic optimization problem, which accounts for and couples all terms: incompressibility, viscosity, surface energy and arbitrary solid constraints.

Our approach builds on the work of Misztal et al. [3] and Erleben et al. [4], who developed a finite element approach to fluid simulation and characterized the solution to the Navier-Stokes equations as a quadratic optimization problem. Our main contribution is the extension of these techniques to multiphase flow. Furthermore, we have addressed numerous implementation issues, essential for significant improvement of the accuracy of the method and reduction of its time complexity. Our contributions include:

- deriving the formulation of and implementing the second-order surface energy approximation (Section 3.3);
- deriving and implementing pressure stabilization through finite volume discretization of the pseudo-compressibility equation (in contrast to the previously used pressure stabilization scheme from Misztal et al. [3], used by Erleben et al. [4], which is not physically correct; see Section 3.4);
- simplifying the formulation of the solid boundary conditions in the model (Section 3.5);
- adapting the method so that it supports multiple, immiscible fluids (Section 3.6).
- designing a preconditioner, which allows us to employ a GPU-based, iterative solver (Section 4);
- suggesting ways to optimize the performance of the DSC method (Section 3.1.1) and the matrix assembly step (Section 5.5);

In contrast to regular grid and level set based approaches, our method offers several significant advantages: the Lagrangian nature of our mesh avoids numerical diffusion that leads to volume loss and

- *M. K. Misztal is with the Niels Bohr Institute, University of Copenhagen, Denmark.*
- *K. Erleben is with the Department of Computer Science, University of Copenhagen, Denmark.*
- *A. Bargteil is with the School of Computing, University of Utah, USA.*
- *J. Fursund is with Industrial Light & Magic.*
- *B. Bunch Christensen is with the Alexandra Institute, Denmark.*
- *J. A. Bærentzen is with the Department of Informatics and Mathematical Modelling, Technical University of Denmark.*
- *R. Bridson is with the Department of Computer Science, University of British Columbia, Canada.*

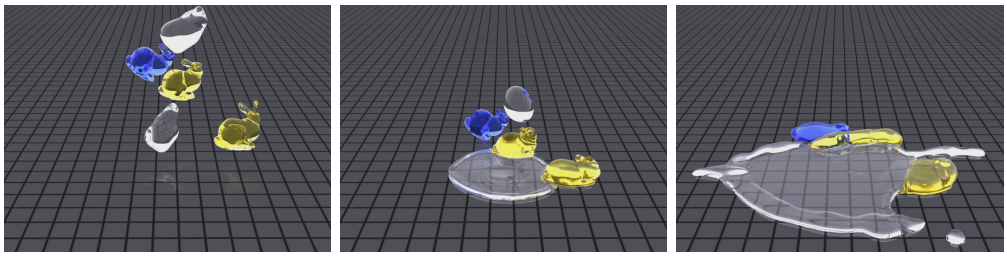

Fig. 1. Multiple, immiscible liquids with different viscosity coefficients and surface tension densities splashing on the bottom of a container. Observe that the simulation has no problem dealing with thin sheets.

excessive perceived viscosity; the unstructured tetrahedral mesh allows us to trivially handle arbitrary, non-grid-aligned solid boundaries, and the explicit representation of interfaces as faces in the mesh allows for accurate treatment of surface tension. In the context of multiphase flow we wish to highlight one additional advantage: the Lagrangian nature of our discretization allows us to optimally track the interfaces between multiple fluids. There is no guesswork in determining what fluids are where and the simulation is greatly simplified because each element is occupied by a single material. This also allows us to assign different values of surface energy density to all pairs of materials. We present several examples of fluid simulations generated using our method, as well as the results of various performance tests and parameter studies.

2 Related Works

The literature concerning fluid simulation in computer graphics is rich, and a proper review of all the state-of-the-art methods would require a study of its own. Most methods are based on regular grids and we refer to [5], [6], [7], [8], [9], [10], [11], [12], [13] for details. The literature on multiphase flow is sparser. Losasso et al. [14] first demonstrated multiple interacting fluids. They used multiple particle level sets to track the various interfaces and introduced averaging rules for handling cells occupied by more than one material. Kim [15] expanded on this approach by introducing regional level sets [16].

Another big group of methods is based on smoothed-particle hydrodynamics [17] and the work by Solenthaler et al. [18] is of particular interest here since it is able to handle multiple materials robustly. In contrast, our method uses an unstructured grid. The work in computer graphics on fluid solvers based on unstructured meshes is sparse. Early work used static unstructured meshes [19], [20]. Dynamic meshes with limited deformation (to preserve mesh quality) were demonstrated by Feldman et al [21]. When mesh quality can no longer be preserved, an entirely new mesh can be generated [22], [23], [24], [25] though this involves a great deal of computation and can lead to undesirable artifacts, such as the smoothing of simulation variables. Alternatively, local mesh improvement operations [3], [26] are computationally more efficient and minimize artifacts. Solid boundaries and two-way coupling have been touched upon [19], [22]. The preferred method for dealing with advection has been the semi-Lagrangian advection method [19] and its generalization to deforming meshes [21] which has been applied in many works [22], [24], [27], [28].

The finite volume method is a popular choice for fluid simulation on unstructured meshes [19], [20], [21], [22], [28]. In particular, Elcott et al. [20] demonstrate a number of desirable, even surprising, properties for incompressible flow simulation with their use of discrete exterior calculus. In contrast, the finite element method [23], [25], [26] is often preferred for plastic and elastic objects. However, its application for fluids is sparse [3], [4]. Mimicking regular grids, many tetrahedral schemes are based on staggered locations of simulation variables [19], [20], [21], [28]. Here the face centers often store the normal velocities and volume centers store pressure values. While this approach has a number of nice properties [20], they have the significant drawback that reconstruction of the full-dimensional velocity field is quite expensive. In contrast, while we do store pressures at the centers of our elements, the velocity field is stored as full-dimensional velocity vectors at the nodes of our mesh.

3 Fluid Simulation Method

In this section we present the complete, finite element discretization of the incompressible fluid motion equations and its formulation in terms of a quadratic optimization problem, which allows us to accurately incorporate non-linear terms (such as surface tension forces) into the model. We also discuss the significant implementation details required for a robust performance: pressure stabilization (in the form of pseudo-compressibility) and preconditioning, which allows us to improve the performance by employing a fast, GPU-based, iterative solver. Finally, we describe how to handle the fluid's interactions with arbitrary solid walls, and how to adapt the method to handle several interacting, immiscible fluids.

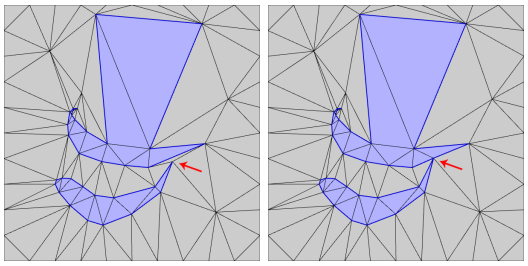

Fig. 2. A 2D example of an interface representation in the DSC method. The embedding mesh can be retessellated in order to accommodate vertex displacement and produce changes in the topology of the interface.

3.1 The Deformable Simplicial Complex Method

Our FEM computations are performed on an unstructured, tetrahedral grid, which is also used by the *deformable simplicial complex* (DSC) method [1], [2] for tracking the fluid's free surface. The DSC method is a recent, Lagrangian method for topology-adaptive deformable interface tracking. It represents the interface explicitly as a piecewise surface (triangle mesh), while the whole embedding space is discretized as well – as a tetrahedral mesh. All tetrahedra are labeled *inside* or *outside* according to their location relative to the interface (the 2D case is shown in Figure 2). Furthermore, they conform to the interface, in the sense that each interface triangle is a common face shared by one *inside* tetrahedron and one *outside* tetrahedron. The interface deformations are produced by iterating interface vertex displacement according to a given velocity field, followed by a mesh improvement step. The main purpose of the mesh improvement step is the removal of low quality tetrahedra produced during vertex advection and reducing the risk of creating inverted tetrahedra in subsequent deformation steps.

The main advantages of the DSC method in the context of fluid simulation include: robust topological adaptivity, low numerical diffusion, available surface mesh representation, which does not change gratuitously between time steps, and the possibility of representing more than two phases (one can use an arbitrary number of tetrahedron labels rather than just two).

Algorithm 1 3DDSC$(M, \mathbf{u})$
$\{M$ is a tetrahedral mesh conforming to the interface$\}$
$\{\mathbf{u}$ is a velocity function for the fluid vertices$\}$

```
1:  t ⇐ 0
2:  while t < t_final do
3:      insert new Steiner vertices
4:      split long interface edges
5:      collapse short interface edges
6:      for each fluid vertex x_i do
7:          compute final vertex position x̃_i ⇐ x_i + u_i Δt
8:      end for
9:      flip interface edges
10:     complete ⇐ false
11:     counter ⇐ 0
12:     while not complete or counter < max_iter do
13:         counter ⇐ counter + 1
14:         complete ⇐ MoveVerticesStep(M, {x̃_i})
15:         relabel valid degenerate tetrahedra
16:         T ⇐ set of tetrahedra adjacent to vertices
            displaced in the previous step
17:         smooth all non-interface vertices in T
18:         if complete and counter ≡ 0 (mod 4) then
19:             ImproveMeshStep(M, T)
20:         else
21:             smooth all outside vertices
22:             reconnection (lines 2-9 in Algorithm 3)
23:             remove degenerate tetrahedra
24:             remove degenerate faces
25:             remove degenerate edges
26:         end if
27:     end while
28:     t ⇐ t + Δt {Δt is a time-step}
29: end while
```

Algorithm 2 MoveVerticesStep$(M, \{\tilde{\mathbf{x}}_i\})$
$\{\{\tilde{\mathbf{x}}_i\}$ is a set of the new positions of the fluid vertices$\}$

```
1:  complete ⇐ true
2:  for each fluid vertex x_i do
3:      if ||x_i − x̃_i|| > 0 then
4:          compute the intersection t_0 of the ray x_i + t ·
            (x̃_i − x_i) with the link of the vertex x_i
5:          if t_0 > 1 then
6:              x_i ⇐ x̃_i
7:          else
8:              move the vertex x_i to the intersection
                point x_i + t_0 · (x̃_i − x_i)
9:              complete ⇐ false
10:         end if
11:     end if
12: end for
13: return complete
```

3.1.1 The DSC Method in 3D

The outline of the 3D DSC method remains very similar to that presented in [2] (see Algorithms 1, 2 and 3). However, we have modified some parts of the algorithm in order to improve the overall performance of the fluid simulation method.

The initial tetrahedral mesh for the simulation is a constrained 3D Delaunay triangulation generated using Tetgen [29]. However, tetrahedral meshes created this way tend to contain nearly-degenerate tetrahedra, so in our mesh improvement algorithms we aim at maximizing the *volume-edge ratio* quality measure [30], [31] $Q(t) = \frac{V(t)}{V_{rms}(t)} = 6\sqrt{2}\frac{V(t)}{e_{\text{rms}}^3}$, where $V(t)$ is the oriented volume of a tetrahedron t and $V_{rms}(t)$ is the volume of a regular tetrahedron with the same root-mean-squared edge length e_{rms} as t. This quality measure is smooth and penalizes all kinds of nearly-degenerate tetrahedra, including slivers.

In the 3D DSC method we perform both volumetric and surface mesh improvement using local operations. The volumetric operations include:

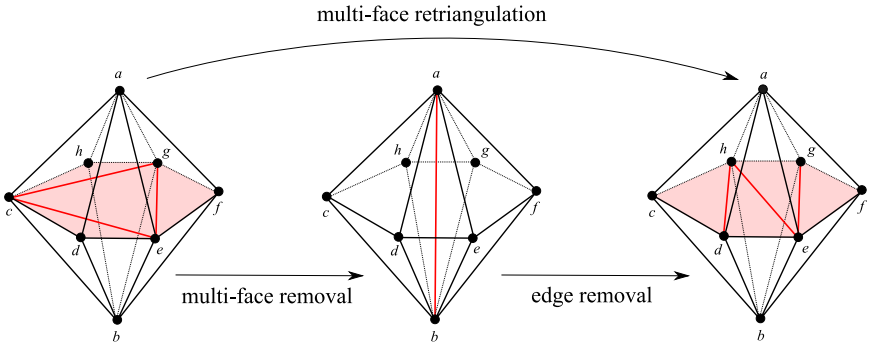

Fig. 3. Topological operations (also called flips or swaps) used for mesh reconnection in 3D DSC. Each operation is performed only if it improves the minimum quality locally.

Algorithm 3 ImproveMeshStep(M, T)

```
smooth all outside vertices in T
for each tetrahedron t ∈ T, such that Q(t) < 0.2 do
    for each non-interface face f of t do
        attempt to remove f using MFRT and MFR
    end for
    for each non-interface edge e of t do
        attempt to remove e using edge removal
    end for
end for
collapse valid non-interface edges
```

- **Mesh quality improvement.** Smart Laplacian smoothing of the non-fluid vertices and *optimization-based smoothing*, which moves the vertex $\mathbf{x}$ in a way that maximizes the minimum quality of its adjacent tetrahedra (this a non-smooth optimization problem, for the details see [32]). Optimization-based smoothing is computationally expensive and we only use it if smart Laplacian smoothing fails to improve the minimum quality Q in the 1-ring of $\mathbf{x}$ above 0.05. We only smooth non-fluid vertices, since arbitrary displacements of the vertices belonging to the computational grid leads to numerical diffusion and inaccuracies.
  Reconnection operations (generalizations of the *edge flip* in the 2D case): *edge remove*, *multi-face remove*, *multi-face retriangulation* (shown in Figure 3, for more details see [33], [34], [35]) if they locally improve the minimum quality. Since the topological operations are computationally expensive, we only perform them for tetrahedra of quality less than 0.2.
- **Interface topology changes.** Topology changes occur when vertices from one component of the interface touch another part of the interface. In this case the two interface parts will be separated only by one or more degenerate tetrahedra. Those tetrahedra can either be re-labelled (switched from *inside* to *outside*, or the other way round) or removed as a part of the degeneracy removal (discussed below).
  We re-label nearly flat tetrahedron t (quality lower than $q_{relabel} = 0.02$) if it is located between two parts of an interface. This is a case when the largest face f of t lies on the interface and the vertex v opposite to f also lies on the interface and its orthogonal projection onto f lies within f's hull. In order to improve the interface mesh we also re-label t if $Q(t) < 10q_{relabel}$ and the change of label would decrease the total surface energy of the interface (acting like "combinatorial surface tension").
- **Detail Control.** Non-interface edges are collapsed if their endpoints do not lie on the interface or the DSC mesh boundary and if the collapse does not locally decrease the minimum quality or if the quality of any tetrahedra affected by this operation does not become lower than $q_{collapse} = 0.30$. Rather than introducing Steiner vertices through non-interface edge splits (as it is done in [2]), we use an optimization based vertex insertion algorithm similar to the one introduced by Klingner and Shewchuk [34]. We use a simplified version of their algorithm which, for any tetrahedron t with quality $Q(t) < 0.2$ attempts to replace a star-shaped set of tetrahedra T containing t, with a new set of tetrahedra connecting the new vertex $\mathbf{x}_{Steiner}$ with each of the faces on the boundary of T. We use the center of the smallest sphere containing t as the initial position of $\mathbf{x}_{Steiner}$, and the star-shaped set T is determined using a graph cut optimization algorithm, described in detail in [34].
- **Degeneracy removal.** *Tetrahedra*: if a tetrahedron's vertices are nearly coplanar (e.g. if the tetrahedron's quality is smaller than a value $q_{min} = 0.01$) its largest face is found and a tetrahedron removal strategy is chosen accordingly to the position of its opposite vertex by "flattening" the tetrahedron and replacing it with a set of 2, 3 or 4 triangles. *Faces*: if a face contains an angle smaller than θ_{min}, $\cos\theta_{min} = 0.998$, it can be either a *cap* or a *needle*. If the ratio of the longest

edge to the second longest edge in the face is greater than 1.03, the longest edge is split at the projection of the vertex opposite to it and the edge connecting the new vertex and the cap tip is collapsed; otherwise, the face is a needle and the edge opposite to the smallest angle is collapsed (both collapses are performed if they produce a valid mesh). *Edges*: degenerate non-interface edges are removed during the edge collapse step. Degenerate interface edges are removed during the process of surface energy improvement described below.

The free surface of a fluid changes dramatically during the simulation and our fluid simulation method would quickly break down had we not kept the surface mesh quality sufficiently high. Moreover, the quality of the embedding tetrahedral mesh depends on the shape and size of the interface mesh triangles. We improve the quality of the surface mesh using the following local operations:

- **Surface mesh quality improvement**. Null-space smoothing [36]: moving each interface, manifold vertex $\mathbf{x}$ in the null space of its local quadric metric tensor. This way the smoothing does not change the geometry of the interface mesh. We update the velocity at the each vertex by applying linear interpolation of the *old* velocity field (before smoothing) at the new position of $\mathbf{x}$.
  Edge flip of an interface, non-manifold edge e: performed if e's adjacent interface triangles do not fulfill 2D Delaunay criterion, if e is not a feature edge (measured by the angle θ_{flip} between the normals to its adjacent faces, $\theta_{flip} < 5^\circ$). Requires performing edge removal (reconnection) operation in the embedding tetrahedral mesh.
- **Detail control**. Edge split for an interface edge e longer than $2.5e_{avg}$, where e_{avg} is the average edge length in the original surface mesh. The velocity at the new vertex is calculated as the average of the velocities at the endpoints of e.
  Edge collapse for an interface edge e shorter than $e_{collapse} = 0.75e_{avg}$. It is important to keep this threshold value high, in order to prevent the mesh size from growing uncontrollably. However, in order to prevent this operation from changing the geometry of the interface significantly, we have introduced a new criterion (absent in [2]). For each of two possibilities of collapsing e, we compute the volume difference they yield. We select the one that yields smaller absolute volume difference, and if it is less than $0.02e^3_{collapse}$ we perform the collapse.

An apparent downside of the 3D DSC method is the number of parameters required to condition the local mesh improvement operations. In principle, the method requires fine tuning of those parameters in order to achieve optimal performance, depending on the application. While changing their values might not have a direct influence on the robustness of the method, it can significantly affect the accuracy and computation time. On the other hand, this also means that the user has much control over the method's performance and capacity to adjust it in order to achieve the desirable results.

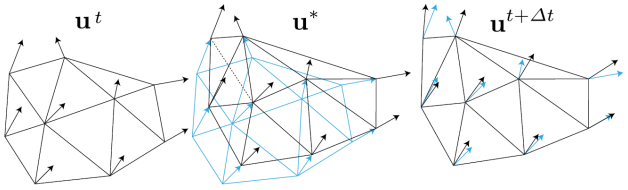

Fig. 4. Schematic view of a single iteration of our fluid simulation method. We begin with an initial velocity field $\mathbf{u}^t$ respecting the continuity constraint (left). Then we move the grid according to this velocity field, advect the velocity values and possibly re-tessellate the mesh (dotted line) in order to accommodate the displacements or improve its quality (center). The new velocity field $\mathbf{u}^*$ might violate the continuity constraint. In order to fix that, we solve the discretized version of the Poisson equation and obtain the final velocity field $\mathbf{u}^{t+\Delta t}$ (right).

In summary, we have described the 3D deformable simplical complex method, adapt for the requirements of fluid simulation. Our main new contributions to the method include: the Steiner vertex insertion routine and the condition for the interface edge collapse.

3.2 Fluid Simulation as a Quadratic Optimization Problem

We treat the tetrahedra contained in each fluid volume as conforming, linear elements. We sample the velocity field $\mathbf{u}(x, y, z)$ at each vertex of the mesh $\mathbf{x}_i$, $i = 1, \ldots, N_V$, where N_V is the total number of vertices in the mesh. We denote the vector of all velocity samples as $\mathbf{u} = \begin{bmatrix} \mathbf{u}_1^T & \mathbf{u}_2^T & \ldots & \mathbf{u}_{N_V}^T \end{bmatrix}^T$, where $\mathbf{u}_i = \mathbf{u}(\mathbf{x}_i)$. We are using a *staggered* grid, meaning that the pressure field is discretized at tetrahedra: $\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \ldots & p_{N_T} \end{bmatrix}^T$, where N_T is the number of elements (tetrahedra) occupied by the fluid. The optimization-based fluid simulation method is a fractional step method. In the first step we perform (forward Euler) vertex advection according to the current sampled velocity field $\mathbf{u}^t$

$$\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \mathbf{u}_i^t \Delta t \tag{1}$$

and we advect the velocity values along with vertices obtaining an intermediate velocity field $\mathbf{u}^*$ which might violate the new, discretized continuity constraint. We fix that in the second step by solving the finite-element discretization of the fluid motion equations in the form of an optimization problem,

which determines the final velocity field $\mathbf{u}^{t+\Delta t}$ of the fluid (the details of the discretization are presented in Section 3.3)

$$\mathbf{u}^{t+\Delta t} = \arg\min_{\mathbf{u}} \quad \frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} + \mathbf{u}^T\mathbf{b} \tag{2}$$

subject to

$$\mathbf{P}^T\mathbf{u} = \mathbf{0} \tag{3}$$

where $\mathbf{A}$ accounts for inertia, viscosity and surface tension, $\mathbf{b}$ contains the effect of the advection and external force densities like gravity, and $\mathbf{P}$ is the gradient operator. The Karush–Kuhn–Tucker (KKT) conditions for this problem read

$$\underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{P}\\ \mathbf{P}^T & \mathbf{0}\end{bmatrix}}_{\mathbf{K}}\begin{bmatrix}\mathbf{u}\\ \lambda\end{bmatrix} = \begin{bmatrix}-\mathbf{b}\\ \mathbf{0}\end{bmatrix} \tag{4}$$

where λ are the Lagrange multipliers and correspond to $-\Delta t\mathbf{p}$ — the pressure field multiplied by the time step size. Observe that solving this problem fully couples the velocity and pressure fields, unlike the projection method. The $\mathbf{K}$ matrix is named the KKT matrix and is known to be symmetric indefinite [37].

3.3 Navier-Stokes Equation Discretization

The motion of a Newtonian fluid is governed by the Navier-Stokes equation,

$$\rho\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\cdot\mathbf{T} + \mathbf{f}, \qquad \mathbf{x}\in\mathcal{V}_{\text{fluid}} \tag{5}$$

where $\mathcal{V}_{\text{fluid}}\subset\mathbf{R}^3$ is the volume of the fluid, ρ is the mass density, $\mathbf{u}$ is the unknown velocity field, and $\mathbf{T}$ is the Newtonian stress tensor:

$$\mathbf{T} = -p\mathbf{I}_{3\times 3} + \mu\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right), \tag{6}$$

where p is the pressure field, μ is the dynamic viscosity coefficient, and $\mathbf{f}$ is an external force term (for example gravity). We assume constant mass density which yields a continuity constraint in the form of incompressibility,

$$\nabla\cdot\mathbf{u} = 0, \qquad \mathbf{x}\in\mathcal{V}_{\text{fluid}}. \tag{7}$$

Erleben et al. [4] show that the weak formulation of this system for a tetrahedral mesh with a staggered grid layout is as follows:

$$\begin{aligned}\mathbf{M}\frac{\partial\mathbf{u}}{\partial t} - \mathbf{B}\mathbf{f} - \mathbf{P}\mathbf{p} + \mathbf{D}\mathbf{u} &= \mathbf{0},\\ \mathbf{P}^T\mathbf{u} &= \mathbf{0},\end{aligned}$$

where $\mathbf{f} = \begin{bmatrix}\mathbf{f}_1^T & \mathbf{f}_2^T & \dots & \mathbf{f}_{N_V}^T\end{bmatrix}$ and

$$\begin{aligned}\mathbf{M}_{ij} &= \mathbf{I}_{3\times 3}\int_{\mathcal{V}_{\text{fluid}}}\rho\phi_i\phi_j dV,\\ \mathbf{B}_{ij} &= \mathbf{I}_{3\times 3}\int_{\mathcal{V}_{\text{fluid}}}\phi_i\phi_j dV,\\ \mathbf{D}_{ij} &= \int_{\mathcal{V}_{\text{fluid}}}\mu\left(\nabla\phi_i^T\nabla\phi_j\mathbf{I}_{3\times 3} + \nabla\phi_i\nabla\phi_j^T\right)dV,\\ \mathbf{P}_{jk} &= \int_{\mathcal{V}_k}\nabla\phi_j dV,\end{aligned}$$

where $i,j = 1,2,\dots,N_V$, $k = 1,2,\dots,N_T$ and $\mathcal{V}_k$ is the volume of the k^{th} tetrahedron. The shape functions $\phi_i : \mathbf{R}^3\mapsto\mathbf{R}$ fulfill the condition

$$\phi_i(\mathbf{x}_j) = \begin{cases}1 & \text{if } i=j\\ 0 & \text{if } i\neq j\end{cases} \tag{8}$$

and are piecewise linear over each element, which allows us to evaluate the matrices above analytically. We apply the finite difference method to discretize Equation 8, by substituting $\frac{\partial\mathbf{u}}{\partial t}\approx\frac{1}{\Delta t}\left(\mathbf{u}^{t+\Delta t}-\mathbf{u}^*\right)$ and, by choosing an implicit scheme for stability we obtain the following system of linear equations

$$\begin{aligned}\mathbf{A}\mathbf{u}^{t+\Delta t} + \mathbf{b} + \mathbf{P}\lambda &= 0, &(9)\\ \mathbf{P}^T\mathbf{u}^{t+\Delta t} &= \mathbf{0}, &(10)\end{aligned}$$

where $\lambda = -\Delta t\mathbf{p}$, $\mathbf{A} = \mathbf{M} + \Delta t\mathbf{D}$, and $\mathbf{b} = -\mathbf{M}\mathbf{u}^* + \Delta t\mathbf{B}\mathbf{f}$. Solving this equation is equivalent to solving the quadratic optimization problem (Equation 2), since its first order optimality conditions are equivalent to Equations 9 and 10. We are interested in this perspective because it allows us to incorporate nonlinear terms into the model, in particular surface tension forces. Adding this term in the form of body forces yields lower accuracy and leads to a stringent stability time step restriction for surface-tension dominated flows. Instead, we add the surface energy term $U(\mathbf{x})$ to our objective function

$$\frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} + \mathbf{u}^T\mathbf{b} + U(\mathbf{x}+\Delta t\mathbf{u}). \tag{11}$$

We use a second-order Taylor series approximation for $U(\mathbf{x}+\Delta t\mathbf{u})$

$$U(\mathbf{x}+\Delta t\mathbf{u}) \approx U(\mathbf{x}) + \Delta t\nabla U\mathbf{u} + \frac{1}{2}\Delta t^2\mathbf{u}^T\nabla\nabla U\mathbf{u},$$

which leads us to another quadratic optimization problem in the standard form with

$$\begin{aligned}\mathbf{A}' &= \mathbf{A} + \Delta t^2\nabla\nabla U, &(12)\\ \mathbf{b}' &= \mathbf{b} + \Delta t\nabla U. &(13)\end{aligned}$$

Surface energy is proportional to the free surface area A of the fluid $U(\mathbf{x}) = \sigma A(\mathbf{x})$. The constant of proportionality σ is called the *surface energy density* and it is a material constant with different values on contact surfaces between each pair of phases (liquid, gaseous and solid) in the system. In order to evaluate the gradient and the Hessian of the energy density (∇U and $\nabla\nabla U$), we need to find the gradient and the Hessian of the area for each interface triangle. We can find symmetric formulas for those by applying a Taylor approximation to Heron's formula for the area A_t of a triangle t with vertices $\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k$. Let us denote $\mathbf{e}_\alpha = \mathbf{x}_\gamma - \mathbf{x}_\beta$, where (α,β,γ) is an even permutation of (i,j,k) and $e_\alpha = \|\mathbf{e}_\alpha\|$. Lengthy calculations lead to the following results

$$\nabla_\alpha A_t = \frac{(e_\alpha^2 - e_\beta^2 + e_\gamma^2)\mathbf{e}_\beta - (e_\alpha^2 + e_\beta^2 - e_\gamma^2)\mathbf{e}_\gamma}{8A_t}, \tag{14}$$

and

$$\begin{aligned}\nabla_\alpha \nabla_\alpha A_t &= \frac{2e_\alpha^2 \mathbf{I} - 2\mathbf{e}_\alpha \mathbf{e}_\alpha^T}{8A_t} - \frac{(\nabla_\alpha A_t)(\nabla_\alpha A_t)^T}{A_t}, \\ \nabla_\alpha \nabla_\beta A_t &= \frac{(e_\gamma^2 - e_\alpha^2 - e_\beta^2)\mathbf{I} - \mathbf{e}_\gamma \mathbf{e}_\gamma^T + \mathbf{e}_\alpha \mathbf{e}_\alpha^T + \mathbf{e}_\beta \mathbf{e}_\beta^T}{8A_t} \\ &\quad - \frac{(\nabla_\alpha A_t)(\nabla_\beta A_t)^T}{A_t},\end{aligned}$$

where ∇_α is the gradient operator with respect to the position of the vertex $\mathbf{x}_\alpha$. For the details on the formulas derivation see Appendix A. Note that Equation 14 is equivalent to the *cotangent formula* [38], commonly used in discrete exterior calculus. A comparison of the fluid simulation results using first-order and second-order surface energy approximations is presented in Section 5.2.

We have presented a finite element discretization of the Navier-Stokes equation, formulated in the form of a quadratic optimization problem, which fully couples the pressure and velocity fields and allows us to accurately include surface tension forces in our model.

3.4 Pressure Stabilization

In some cases, the matrix $\mathbf{P}$ might not have full column rank, making the KKT matrix singular. In the finite element literature this is referred to as *locking*. To circumvent this problem we add a stabilization term to the KKT system.

We apply the idea of pseudo-compressibility [39] to stabilize the Navier–Stokes equations. There are different versions of this class of pseudo-compressibility methods. However, the version we use replaces the continuity constraint $\nabla \cdot \mathbf{u} = 0$ with

$$\nabla \cdot \mathbf{u} - \frac{\varepsilon}{\rho} \nabla^2 p = 0 \tag{15}$$

where ε is termed the *stabilization parameter* and is related to the time step one is using. Shen [39] suggests using $\varepsilon \approx \Delta t$.

We can discretize this modified continuity equation using a finite volume method

$$0 = \int_{\mathcal{V}_{\text{fluid}}} \left(\nabla \cdot \mathbf{u} - \frac{\Delta t}{\rho} \nabla^2 p \right) dV \approx \mathbf{P}^T \mathbf{u} - \Delta t \mathbf{S} \mathbf{p}. \tag{16}$$

The formulation of the matrix $\mathbf{P}$ has already been shown in Section 3.3. In order to evaluate the second term, we split the volume integral into the sum of integrals over each tetrahedron and apply Gauss' theorem, which yields

$$\frac{\Delta t}{\rho} \int_{\mathcal{V}_{\text{fluid}}} (\nabla^2 p)\, dV = \frac{\Delta t}{\rho} \sum_k \sum_l \int_{A_{kl}} (\nabla p \cdot \mathbf{n}_{kl})\, dA,$$

where A_{kl} is a face of the k^{th} tetrahedron, shared with the l^{th} tetrahedron, and $\mathbf{n}_{kl}$ is the normal vector to A_{kl}. Now, assuming that the pressure field is discretized at the barycenters of the elements in our mesh we can approximate the term $\nabla p \cdot \mathbf{n}_{kl}$ on A_{kl}.

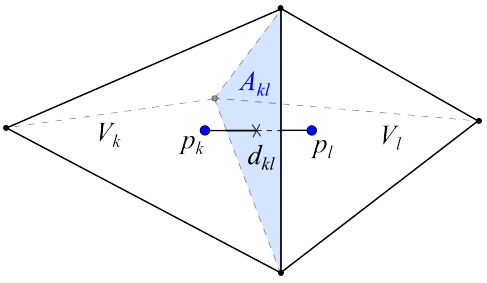

Fig. 5. The distance d_k from the barycenter of a tetrahedron k to its face A_{kl} is four times smaller than its height h_k relative to that face. The volume of this tetrahedron $V_k = \frac{1}{3} h_k A_{kl}$, hence $d_k = \frac{3V_k}{4A_{kl}}$. Analogously, $d_l = \frac{3V_l}{4A_{kl}}$. From this, we have that the distance between the barycenter, in the direction orthogonal to A_{kl} equals $d_{kl} = d_k + d_l = \frac{3}{4} \frac{V_k + V_l}{A_{kl}}$.

We do it by evaluating the term $p(\mathbf{x} + d\mathbf{n})$ using Taylor approximation

$$p(\mathbf{x} + d\mathbf{n}) \approx p(\mathbf{x}) + (\nabla p(\mathbf{x}) \cdot \mathbf{n})\, d. \tag{17}$$

From this

$$\nabla p \cdot \mathbf{n}_{kl} \approx \frac{p_l - p_k}{d_{kl}}, \tag{18}$$

where d_{kl} is the distance between the barycenters of the k^{th} and the l^{th} tetrahedra projected onto $\mathbf{n}_{kl}$. This is a good approximation as long as the barycenters of tetrahedra k and l project onto the same point on A_{kj}. Fortunately, the DSC method optimizes the mesh to favor this property. This approximation has been used previously by Chentenez et al. [24], and similar approximations are also used in computational fluid dynamics [40].

We can express the formula shown in Equation 18 using the area of A_{kl} and the volumes of its adjacent tetrahedra

$$d_{kl} = \frac{3}{4} \frac{V_k + V_l}{A_{kl}} \tag{19}$$

(see Figure 5 for the explanation). Hence

$$\frac{\Delta t}{\rho} \int_{\mathcal{V}_{\text{fluid}}} (\nabla^2 p)\, dV \approx \frac{\Delta t}{\rho} \sum_k \sum_l \frac{A_{kl}}{d_{kl}} (p_l - p_k). \tag{20}$$

For the sake of brevity, let us denote

$$\delta_{kl} = \frac{A_{kl}}{d_{kl}} = \frac{4}{3} \frac{A_{kl}^2}{V_k + V_l}. \tag{21}$$

This way we can write the matrix $\mathbf{S}$ as

$$\mathbf{S}_{kl} = \frac{1}{\rho} \begin{cases} \delta_{kl} & \text{if } k \neq l \\ -\sum_{m \neq k} \delta_{km} & \text{if } k = l \end{cases}, \tag{22}$$

where δ_{kl} is given by Equation 21 if tetrahedra k and l share a face, or otherwise equals 0. Such a pressure stabilization term relaxes the incompressibility constraint by allowing limited volume exchange between adjacent tetrahedra, while keeping the total volume of the fluid constant.

We can easily include this term in our KKT system (Equation 4) by replacing the continuity constraint $\mathbf{P}^T\mathbf{u} = \mathbf{0}$ with Equation 16, obtaining

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{23}$$

The comparison of the fluid simulation results using this pressure stabilization scheme and the one by Misztal et al. [3] is presented in Section 5.2.

3.5 Solid Boundaries

In the computer graphics community there are two popular choices of boundary condition equations at the contact surface between the fluid and the solid boundaries. The *free-slip* condition states that at the solid boundaries the normal velocity of the fluid must be 0 (in case the solid wall is static) or must match the normal velocity of the solid. This boundary condition is a popular choice for fluids with low viscosity values. The *no-slip* condition states that at the solid boundaries, the fluid does not move relative to the boundary (its velocity matches that of the solid). This boundary condition is favored when modeling fluids with high-viscosity values. In our experiments, we have been using the former approach, although implementation of a no-slip condition is also possible in our framework.

Let us focus on a static solid wall $\mathcal{W} \subset \mathbf{R}^3$ (including moving solids is straight-forward and only changes the left-hand side part of our KKT system). Let us denote the set of all fluid vertices in contact with the solid boundary as

$$\mathcal{C} = \{k : \mathbf{p}_k \in \partial\mathcal{W}\}, \tag{24}$$

where $\mathbf{p}_k$ is the position of the k^{th} vertex. We may now write the free-slip solid boundary condition for a vertex $k \in \mathcal{C}$ as

$$\mathbf{n}_k^T\mathbf{u}_k = 0 \qquad \forall k \in \mathcal{C}, \tag{25}$$

where $\mathbf{u}_k$ is the fluid's velocity at the k^{th} vertex and $\mathbf{n}_k$ is the normal to the boundary at $\mathbf{p}_k$. Given the velocity field $\mathbf{u} \in \mathbf{R}^{3N_V}$ we may now define the boundary condition at solid walls as

$$\mathbf{C}\mathbf{u} = \mathbf{0} \tag{26}$$

where $\mathbf{C} \in \mathbf{R}^{\|\mathcal{C}\| \times 3N_V}$. Now we may add the solid boundary conditions to our optimization problem as a hard constraint

$$\mathbf{u}^{t+\Delta t} = \arg\min_{\mathbf{u}} \quad \frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} + \mathbf{u}^T\mathbf{b} \tag{27}$$

subject to

$$\mathbf{P}^T\mathbf{u} = \mathbf{0} \tag{28}$$

$$\mathbf{C}\mathbf{u} = \mathbf{0} \tag{29}$$

This results in a KKT-matrix

$$\mathbf{K}' = \begin{bmatrix} \mathbf{A} & \mathbf{P} & \mathbf{C}^T \\ \mathbf{P}^T & \varepsilon\mathbf{S} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{30}$$

This is clearly a symmetric indefinite matrix. The $\mathbf{C}$-matrix has full row rank and therefore the KKT-matrix $\mathbf{K}'$ is nonsingular.

3.6 Multiple phases

One can easily adapt the DSC method so that it handles multiple phases. Instead of having just two labels for tetrahedra (*inside* and *outside*), one can use an arbitrary number of labels, each representing a different phase. This allows us to simulate several, immiscible fluids with different density, surface tension and viscosity values.

Since each tetrahedron in the mesh is occupied by just one fluid, the solver remains essentially unchanged. We apply full-slip boundary conditions on the contact surface between each pair of interacting fluids, meaning that the vertices on the interface between two fluids are given freedom to move in every direction. The discretization of the Navier-Stokes equation presented in Section 3.3 remains valid when we associate different density, viscosity and surface energy density values with different elements. The only part that needs changing is the pressure stabilization term. In order to avoid exchanging volume between two different fluids, we modify the matrix $\mathbf{S}$ (given by Equation 22) as follows

$$\mathbf{S}_{kl} = \frac{1}{\rho_i} \begin{cases} \delta_{kl} & \text{if } k \neq l \\ -\sum_{m \neq k} \delta_{km} & \text{if } k = l \end{cases}, \tag{31}$$

where δ_{kl} is given by Equation 21 if tetrahedra k and l share a face and belong to the same fluid (have the same label i), or otherwise equals 0.

4 Using a Preconditioned Iterative Solver

The KKT system solving step was the main bottleneck in the previous work [4]. The fluid simulation method would spend up to 70% of the computation time solving the linear system using the Cholesky decomposition method. This is why we decided to use an iterative solver in our work. The indefiniteness of the modified KKT-matrix may cause numerical problems when we want to solve our system of linear equations. Ideally we would like to apply a scheme like the Conjugate Gradient (CG) method. However, CG typically does not converge well for indefinite systems.

Typically MINRES, SYMMLQ [41] or GMRES [42] are used instead of CG when dealing with an indefinite matrix. In earlier simulations we used the Generalized Minimum Residual (GMRES) method. It

is similar to CG except it keeps a memory limited local storage of vectors spanning the Krylov space that is being explored [43]. One can find off-the-shelf GPU implementations of GMRES which can boost the performance with almost no programming effort. In later simulation we replaced the GMRES method with the Conjugate Residual (CR) method [44], which takes the symmetry of our KKT matrix into account, improving the overall performance of our fluid simulation method (see Tables 1 and 2). In both cases we use CUSP [45].

The matrix $\mathbf{K}$ is not diagonally dominant, so we can not use the well-known Jacobi preconditioner. Instead, in order to improve the GMRES or CR method's convergence rate, we apply a diagonal approximation of Murphy's block preconditioner [46]. It is based on the idea of using a diagonal version of the Schur complement as a preconditioner

$$\mathbf{P}_{\text{schur}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \left(\mathbf{S} - \mathbf{P}\mathbf{A}^{-1}\mathbf{P}^T\right) \end{bmatrix} \tag{32}$$

The diagonal approximation of this preconditioner would be

$$\mathbf{P}_{\text{diag}} = \begin{bmatrix} \text{diag}\left(\mathbf{A}\right) & \mathbf{0} \\ \mathbf{0} & \text{diag}\left(\mathbf{S} - \mathbf{P}\left(\text{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{P}^T\right) \end{bmatrix} \tag{33}$$

This preconditioner is inexpensive to compute. Furthermore, $\mathbf{P}_{\text{diag}}$ is trivial to invert. Hence we solve the preconditioned system

$$\mathbf{P}_{\text{diag}}^{-1}\mathbf{K}\begin{bmatrix}\mathbf{u}\\ \lambda\end{bmatrix} = \mathbf{P}_{\text{diag}}^{-1}\begin{bmatrix}-\mathbf{b}\\ \mathbf{0}\end{bmatrix}. \tag{34}$$

For a modified KKT-matrix, which includes the solid constraints

$$\mathbf{K}' = \begin{bmatrix} \mathbf{A} & \mathbf{P} & \mathbf{C}^T \\ \mathbf{P}^T & \varepsilon\mathbf{S} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{35}$$

(compare Section 3.5) we modify our preconditioner as follows

$$\mathbf{P}'_{\text{diag}} = \begin{bmatrix} \mathbf{P}_{\text{diag}} & \mathbf{0} \\ \mathbf{0} & \text{diag}\left(-\mathbf{C}^T\left(\text{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{C}\right) \end{bmatrix}. \tag{36}$$

In our experiments this seems to works well in combination with both GMRES (see Figure 6 for convergence plots) and CR.

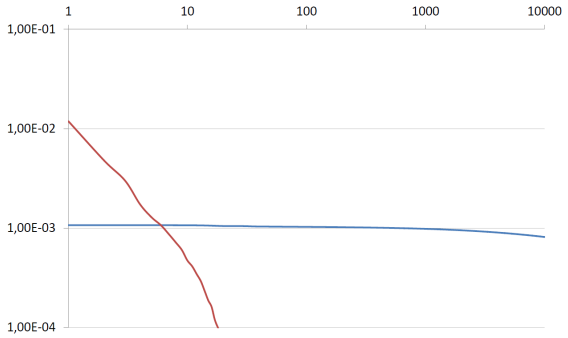

Fig. 6. Typical convergence behavior for a GMRES solver using our preconditioner (red line) and without a preconditioner (blue line) in a flow dominated by surface tension: the horizontal axis shows the number of GMRES iterations and the vertical axis – the relative residual. In this example, preconditioning helps GMRES converge to a desired final residual of 10^{-4} in as few as 20 iterations.

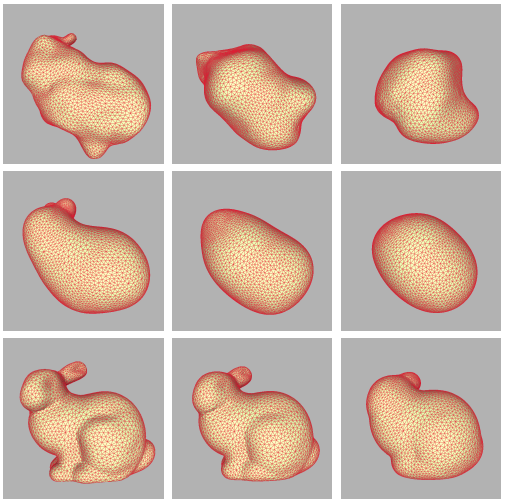

Fig. 7. Stanford bunny model deforming in zero gravity due to the surface tension forces after (from the left to the right) 1, 2 and 3 seconds. The fluid's viscosity, from the top to the bottom: $\mu = 0$ P (the unit of viscosity), $\mu = 0.1$ P and $\mu = 1$ P.

5 Tests and Results

5.1 Viscosity

In order to validate our viscosity model, we have run a simple experiment, in which a Stanford bunny model, given different viscosity coefficient values, deforms freely due to the surface tension force (Figure 7). The results of the experiment follow the intuition: when the viscosity coefficient is low, the fluid volume deforms rapidly, however it takes a long time to lose its kinetic energy and keeps oscillating; as we increase the viscosity coefficient, we introduce more damping — the deformation progresses more slowly and the initial shape smoothly transitions into an oval, and further on — into a sphere.

5.2 Capillary waves

One of the applications requiring both low numerical diffusion and accurate treatment of the surface tension forces is the simulation of capillary waves. The discussion of the problem and benchmark results have been provided by Brochu et al. [12]. We have repeated one of their experiments to see how our method deals with this problem. Our results are in

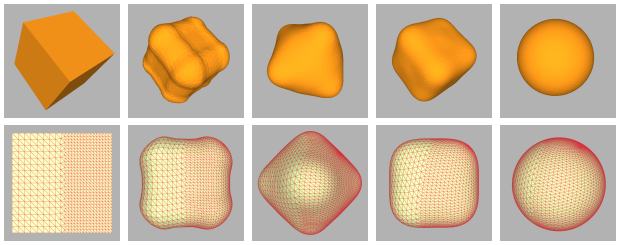

Fig. 8. A uniformly (top row) and non-uniformly (bottom row) tessellated cube in zero gravity deforming due to surface tension forces. Rather than deforming directly into a sphere, the blob of fluid oscillates rapidly between an octahedron-like and a cube-like shape until its kinetic energy dissipates and it becomes spherical. Notice that that non-uniform tessellation of the initial volume of fluid does not affect the behavior of the fluid significantly, nor does it introduce ghost forces.

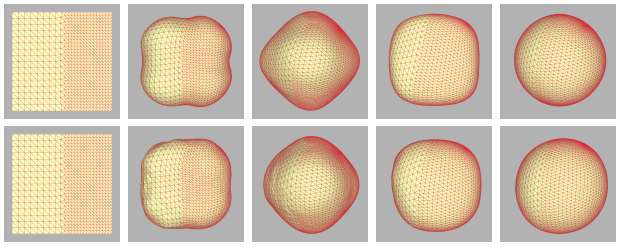

Fig. 9. Capillary waves experiment results with first-order surface energy term (top row) and with the previous pressure stabilization scheme from [3] (bottom row). In the former case, the behavior of the fluid is similar to that presented in Figure 8, however, noticeable asymmetry and slight drift emerge. In the latter case, simulation quality is significantly lower and the capillary waves are not captured correctly.

agreement with previous work (they are presented in Figure 8). Note that the simulation results do not depend significantly on the initial tessellation of the fluid volume. This is the case, however, in the earlier approaches by Misztal et al. [3] and Erleben et al. [4] (as shown in Figure 9). While using the first-order surface energy approximation leads to generally sane results, the free surface of the fluid quickly becomes visibly asymmetric, and the fluid volume begins to drift. Using the pressure stabilization scheme from [3] dramatically deteriorates the simulation quality, introduces ghost forces (causing the drift of the fluid) and practically prevents us from capturing the capillary effects at all.

5.3 Droplet pinch-off

In nearly all fluid animation methods droplet pinch-off is a consequence of disintegrating thin liquid threads or sheets. In the level set based approaches to fluid animation, this usually happens when the scale of those features becomes lower than the resolution of the computational grid. While this usually leads to plausibly looking results, it is hardly physically correct. Particle-based approaches produce droplets when some particles travel beyond the interaction distance from the main (continuous) volume of the liquid. This, again, is not physically accurate, since in the macroscopic scale liquids do not exhibit such discrete behavior.

Unlike those two approaches, the DSC-based approach is much more conservative. It is capable of representing arbitrarily thin features, since all criteria for interface merging or splitting are quality-dependent, rather than scale-dependent. In principle, the interface (free surface of the liquid) splits only when self-collisions occur. That means, in order to understand droplet pinch-off in the DSC-based fluid animation framework, we have to look at the actual physical phenomena leading to droplet pinch-off in real free surface flows.

The break-up of thin liquid threads into droplets and the conditions ruling when this happens are explained by the *Plateau-Rayleigh instability* [47], [48]. The principal factor leading to disintegration is the surface tension, hence, droplet pinch-off cannot be simulated accurately without accurate treatment of the surface tension forces. Naturally occuring perturbations to the free surface of the fluid thread are propagated along the thread as capillary waves. While some of those waves decay in time, some grow at a fast rate. The capillary wave formation also occurs when the cross section of the liquid thread is not circular (due to the surface tension forces trying to minimize the free surface). It has been proven theoretically, that for a vertically falling column of liquid with circular cross-section, droplet pinch off occurs if its wavelength is greater than its circumference. In the general setting, the conditions for droplet pinch-off are more complex and are outside of the scope of this paper, however, they have been a subject of extensive research due to their relevance in various branches of technology, for example ink-jet printing [48]. In Figure 10 we present the results of our simulations displaying the break-up of thin liquid threads.

The case of disintegration of thin liquid sheets is even more complex, however it has been a subject of research within the field of rheology [49]. In principle, it is similar to the disintegration of liquid threads, in the sense that the droplets pinch off from the sheet's boundary, which, in the presence of surface tension forces, behaves similarly to a liquid thread. The discussion of this phenomenon is outside the scope of this paper, however, we present the results of a simulation displaying disintegration a liquid sheet in our framework (Figure 11).

5.4 Other experiments

In order to verify our model, we have performed a "crown" experiment in which a spherical droplet

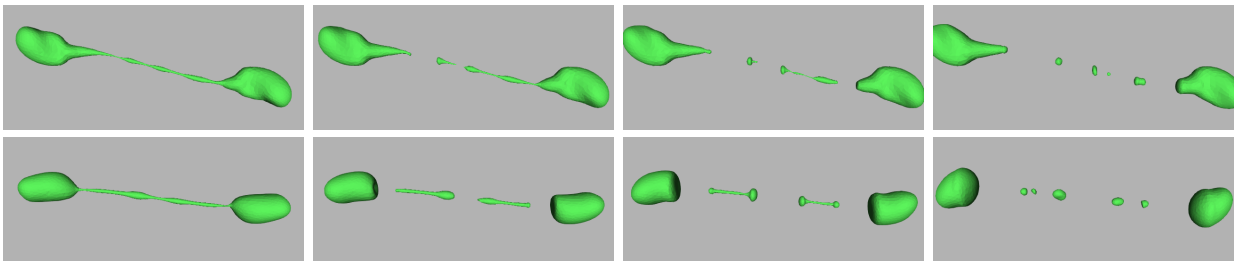

Fig. 10. The Plateau-Rayleigh instability leading to disintegration of liquid threads produced by an oblique collision of two spherical droplets due, for two different surface tension values: $\sigma_1 = 2\ \mathrm{dyn/cm}$ (top row) and $\sigma_2 = 10\ \mathrm{dyn/cm}$ (bottom row).

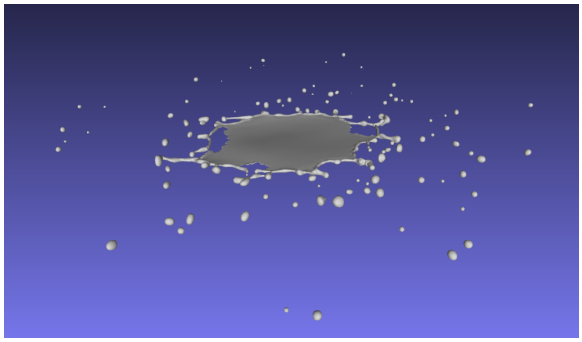

Fig. 11. Disintegration of a liquid sheet produced by a head-on collision of two spherical droplets. Droplets pinch off from the boundary of the sheet due to the surface tension forces.

falls into a shallow layer of liquid. The results of the simulation are realistic, as shown in Figure 12. Observe that proper handling of thin sheets of fluid comes naturally in our method.

Figures 1 and 13 present the results of our experiments with multiple immiscible fluids: in particular water and oil. Each type of contact between fluid, solid and gaseous phases is assigned different surface energy densities. We observe qualitatively different behavior in the different phases.

5.5 Performance

We have run all our experiments on a machine with an Intel® Core™ i7 CPU X 980 3.33GHz with an NVIDIA® Geforce™ GTX580 GPU.

The statistics of our simulations before performance optimization are presented in Table 1. The timings are comparable to other finite element based simulation methods [26]. By applying an iterative solver, we have significantly decreased the time spent on solving the KKT system. The DSC method's mesh improvement functionality seems to work robustly, particularly for single phase simulations, where it allows us to keep most of the dihedral angles in the range $10^\circ - 160^\circ$ except for the times when collisions occur. Those times, unfortunately, tend to introduce low quality tetrahedra which might not be removed immediately. While the fluid simulation method seems to deal with such elements rather well, they negatively affect the performance during the advection step. We are planning to address this issue by investigating more sophisticated mesh refinement schemes.

In Table 2 we present the statistics of some of our simulations after performance optimization, including the improved interface edge collapse routine (see Section 3.1.1), using the CR solver rather than GMRES (see Section 4) and parallelizing the matrix assembly step. The latter has been done by rewriting the matrix assembly code and using the OpenMP [50] API, utilizing all six cores of the CPU (12 threads). This has lead to significant performance improvement, decreasing the computation time per iteration by about 50%. The improvement in the advection step timings in the new results is a consequence of several minor changes and improvements in the implementation of the DSC method.

6 Summary and Discussion

The distinguishing characteristic of our scheme is that it is Lagrangian with an explicit interface representation, yet also volumetric, using a single irregular grid for both simulation as well as tracking and handling collisions of parts of the interface. In this work, we have demonstrated that the method can deal with multiphase flows and that the qualitative behavior of the simulated fluid is as expected.

Our new pressure stabilization strategy resulted in lower numerical diffusion than in [3], [4], allowing us to capture the capillary waves correctly and making our simulations of surface tension dominated flows on par with the state-of-the-art methods [12], [51]. Thin sheets are handled accurately without the need for any special treatment. Furthermore, our new pressure stabilization scheme made the method insensitive to the mesh element size, removing the problem of ghost forces present in earlier works when the initial tessellation of the fluid volume is non-uniform. This way, we have opened the doors for an adaptive-resolution, multi-scale fluid simulation using our framework.

Compared to [3], we have also improved the treatment of solid boundaries. The presented formulation

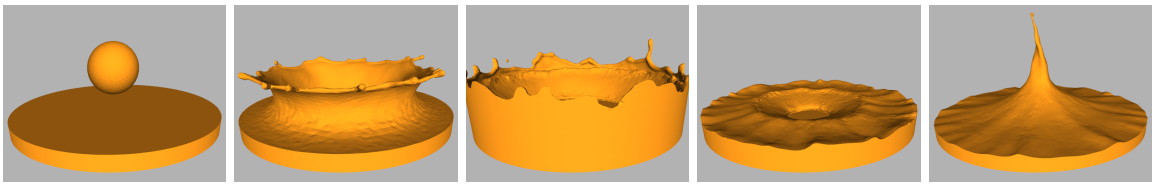

Fig. 12. A spherical droplet splashing in a cylindrical container with a shallow layer of water, producing a "crown". Observe that our method does not have any problems with handling thin layers of fluid.

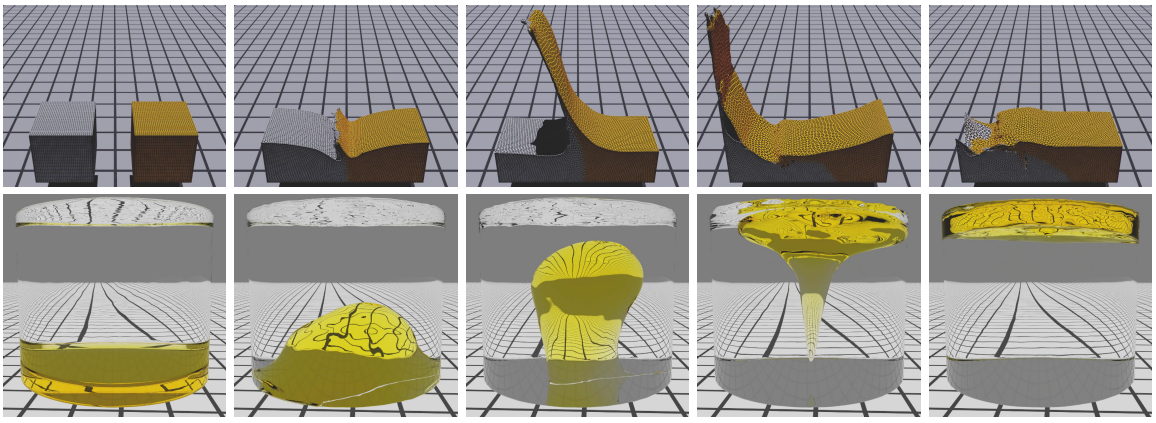

Fig. 13. The results of our two-phase experiments with water and oil: the "double dam breaking" experiment in which the collision of the volumes of oil and water produces a rapidly moving jet (top); the movement of a small volume of oil in water due to the difference in densities (bottom).

TABLE 1
Simulation statistics before performance optimization: the initial number of tetrahedra of the embedding mesh (excluding tetrahedra labeled *outside* which are not used for computation); the initial/max/median number of tetrahedra of the computational mesh; the initial/max/median number of the surface elements; the average timings of each step of the simulation: matrix assembly, solving the linear system using the GMRES method, advection step using the DSC method (including mesh improvement); the total average timing of an iteration.

example	#tets initial/max/median	#triangles initial/max/median	average time per iteration (seconds) assembly	GMRES	advection	total
CUBE N.U.	20345 / 20345 / 17403	7616 / 7616 / 7610	6.41	3.48	3.87	13.9
OIL/WATER	42319 / 42319 / 24538	8832 / 9532 / 9040	8.91	6.57	4.19	19.9
TEASER	24770 / 92893 / 43983	11840 / 37089 / 20575	22.0	11.2	14.4	48.1
CROWN	33810 / 94789 / 81983	15104 / 38054 / 32160	32.7	19.6	19.8	72.8
DAM BREAKING	84852 / 108518 / 98722	33370 / 42736 / 38291	40.2	21.4	23.1	85.6

works well with the iterative linear system solver and simplifies adding moving solids to the model in the future, in contrast to the approach presented in [3]. The use of a preconditioned iterative solver allows us to decrease the amount of time spent on solving the linear system, which was the bottleneck in [4].

In the future we would like to investigate different mesh refinement schemes, which would allow us to improve the computational mesh quality when changes in the surface mesh topology take place. We would also like to explore the applicability of our method in simulating interactions between fluids and deformable solid bodies.

ACKNOWLEDGMENTS

This work was funded by a grant from the Danish Agency for Science, Technology and Innovation and partially by NSF grants IIS-1249756 and CNS-0855167. We would like to thank anonymous reviewers for their valuable feedback.

REFERENCES

[1] M. K. Misztal, "Deformable simplicial complexes," Ph.D. dissertation, Technical University of Denmark (DTU), Denmark, 2010.

[2] M. K. Misztal and J. A. Bærentzen, "Topology adaptive interface tracking using the deformable simplicial complex," *ACM Trans. Graph.*, vol. 31, no. 3, pp. 24:1–24:12, Jun. 2012.

TABLE 2
Selected simulation statistics after performance optimization, including improved edge collapse, parallelization of the matrix assembly and use of the CR solver.

example	#tets initial/max/median	#triangles initial/max/median	average time per iteration (seconds) assembly	CR solver	advection	total
CUBE N.U.	20349 / 20349 / 19247	7616 / 7916 / 7715	2.01	3.01	2.32	7.4
BUNNY/ARMADILLO	43252 / 43252 / 40825	18572 / 18527 / 17987	4.56	6.25	5.47	16.5

[3] M. K. Misztal, R. Bridson, K. Erleben, J. A. Bærentzen, and F. Anton, "Optimization-based fluid simulation on unstructured meshes," in *Proc. of the 7th Workshop on Virtual Reality Interactions and Physical Simulations, VRIPHYS*, 2010, pp. 11–20.

[4] K. Erleben, M. K. Misztal, and J. A. Bærentzen, "Mathematical foundation of the optimization-based fluid animation method," in *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symp. on Comp. Animation*, 2011, pp. 101–110.

[5] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, no. 5, pp. 471–483, 1996.

[6] ——, "Modeling the motion of a hot, turbulent gas," in *SIGGRAPH '97: Proc. of the 24th annual conference on Comp. graphics and interactive techniques*, 1997, pp. 181–188.

[7] J. Stam, "Stable fluids," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Comp. graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128.

[8] N. Foster and R. Fedkiw, "Practical animation of liquids," in *SIGGRAPH '01: Proc. of the 28th annual conference on Comp. graphics and interactive techniques*. ACM, 2001, pp. 23–30.

[9] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *SIGGRAPH '01: Proc. of the 28th annual conference on Comp. graphics and interactive techniques*. New York, NY, USA: ACM, 2001, pp. 15–22.

[10] R. Bridson and M. Müller-Fischer, "Fluid simulation: Siggraph 2007 course notes," in *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*. New York, NY, USA: ACM, 2007, pp. 1–81.

[11] R. Bridson, *Fluid Simulation*. Natick, MA, USA: A. K. Peters, Ltd., 2008.

[12] T. Brochu, C. Batty, and R. Bridson, "Matching fluid simulation elements to surface geometry and topology," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 47:1–47:9, Jul. 2010.

[13] A. McAdams, E. Sifakis, and J. Teran, "A parallel multigrid poisson solver for fluids simulation on large grids," in *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symp. on Comp. Animation*, 2010, pp. 65–74.

[14] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 812–819, 2006.

[15] B. Kim, "Multi-phase fluid simulations using regional level sets," *ACM Trans. Graph.*, vol. 29, pp. 175:1–175:8, December 2010.

[16] W. Zheng, J.-H. Yong, and J.-C. Paul, "Simulation of bubbles," in *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comp. animation*, ser. SCA '06. Eurographics Association, 2006, pp. 325–333.

[17] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual review of astronomy and astrophysics*, vol. 30, pp. 543–574, 1992.

[18] B. Solenthaler, J. Schläfli, and R. Pajarola, "A unified particle model for fluid solid interactions," *Comput. Animat. Virtual Worlds*, vol. 18, pp. 69–82, 2007.

[19] B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 904–909.

[20] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Trans. Graph.*, vol. 26, no. 1, p. 4, 2007.

[21] B. E. Feldman, J. F. O'Brien, B. M. Klingner, and T. G. Goktekin, "Fluids in deforming meshes," in *SCA '05: Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comp. animation*. ACM, 2005, pp. 255–259.

[22] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien, "Fluid animation with dynamic meshes," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 820–825, 2006.

[23] A. W. Bargteil, C. Wojtan, J. K. Hodgins, and G. Turk, "A finite element method for animating large viscoplastic flow," *ACM Trans. Graph.*, vol. 26, July 2007.

[24] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in *SCA '07: Proc. of the 2007 ACM SIGGRAPH/Eurographics Symp. on Comp. animation*, 2007, pp. 219–228.

[25] C. Wojtan and G. Turk, "Fast viscoelastic behavior with thin features," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*. New York, NY, USA: ACM, 2008, pp. 1–8.

[26] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien, "Dynamic local remeshing for elastoplastic simulation," *ACM Trans. Graph.*, vol. 29, pp. 49:1–49:11, July 2010.

[27] N. Chentanez, T. G. Goktekin, B. E. Feldman, and J. F. O'Brien, "Simultaneous coupling of fluids and deformable bodies," in *SCA '06: Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comp. animation*, 2006, pp. 83–89.

[28] J. D. Wendt, W. Baxter, I. Oguz, and M. C. Lin, "Finite volume flow simulations on arbitrary domains," *Graph. Models*, vol. 69, no. 1, pp. 19–32, 2007.

[29] H. Si, "Tetgen, a quality tetrahedral mesh generator and three-dimensional delaunay triangulator, v1.3 user's manual," WIAS, Tech. Rep., 2004.

[30] V. N. Parthasarathy, C. M. Graichen, and A. F. Hathaway, "A comparison of tetrahedron quality measures," *Finite Elements in Analysis and Design*, vol. 15, no. 3, pp. 255–261, 1994.

[31] A. Liu and B. Joe, "Relationship between tetrahedron shape measures," *BIT Numerical Mathematics*, vol. 34, no. 2, pp. 268–287, 1994.

[32] L. A. Freitag, M. Jones, and P. Plassmann, "An efficient parallel algorithm for mesh smoothing," in *Proc. of the Fourth International Meshing Roundtable*, 1995, pp. 103–112.

[33] L. A. Freitag and C. Ollivier-Gooch, "Tetrahedral mesh improvement using swapping and smoothing," *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 3979–4002, 1997.

[34] B. M. Klingner and J. R. Shewchuk, "Agressive tetrahedral mesh improvement," in *Proc. of the 16th International Meshing Roundtable*, Oct. 2007, pp. 3–23.

[35] M. K. Misztal, J. A. Bærentzen, F. Anton, and K. Erleben, "Tetrahedral mesh improvement using multi-face retriangulation," in *Proceedings of the 18th International Meshing Roundtable*, Oct. 2009, pp. 539–556.

[36] X. Jiao, "Face offsetting: A unified approach for explicit moving interfaces," *J. Comput. Phys.*, vol. 220, no. 2, pp. 612–625, 2007.

[37] J. Nocedal and S. J. Wright, *Numerical optimization*, ser. Springer Series in Operations Research. New York: Springer-Verlag, 1999.

[38] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[39] J. Shen, "Pseudo-compressibility methods for the unsteady incompressible navier-stokes equations," in *Proc. of the 1994 Beijing Symp. on Nonlinear Evolution Equations and Infinite Dynamical Systems*. ZhongShan University Press, 1997, pp. 68–78.

[40] H. K. Versteeg and W. Malalasekera, *An introduction to com-*

putational fluid dynamics: the Finite Volume method. Longman Scientific and Technical, 1995.
[41] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 12, no. 4, pp. 617–629, 1975.
[42] Y. Saad and M. H. Schultz, "Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, Jul. 1986.
[43] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.
[44] D. G. Luenberger, "The conjugate residual method for constrained minimization problems," *SIAM J. Numer. Anal.*, vol. 7, no. 3, 1970.
[45] N. Bell and M. Garland, "Cusp: Generic parallel algorithms for sparse matrix and graph computations," 2010, version 0.1.0.
[46] K. K. Phoon, K. C. Toh, S. H. Chan, and F. H. Lee, *Computational Fluid and Solid Mechanics 2003: Second Mit Conference 2003*. Elsevier Science Ltd, 2003, vol. 1, ch. A Generalised Jacobi Preconditioner for Finite Element Solution of Large-Scale Consolidation Problems, pp. 573–577.
[47] D. T. Papageorgiou, "On the breakup of viscous liquid threads," *Physics of Fluids*, vol. 7, no. 7, p. 1529, 1995.
[48] J. Eggers, "Nonlinear dynamics and breakup of free-surface flows," *Rev. Mod. Phys.*, vol. 69, pp. 865–930, Jul 1997.
[49] G. Taylor, "The dynamics of thin sheets of fluid. iii. disintegration of fluid sheets," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 253, no. 1274, pp. pp. 313–321, 1959.
[50] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald, *Parallel Programming in OpenMP*. Morgan Kaufmann, 2000.
[51] N. Thürey, C. Wojtan, M. Gross, and G. Turk, "A multiscale approach to mesh-based surface tension flows," in *SIGGRAPH '10: ACM SIGGRAPH 2010 papers*. New York, NY, USA: ACM, 2010, pp. 1–10.

Marek Krzysztof Misztal received his PhD degree in computer science from the Technical University of Denmark in 2011. He is currently a postdoctoral fellow at the Niels Bohr Institute, University of Copenhagen. His research interests include geometry processing, physics-based simulation and scientific computing.

Kenny Erleben received his PhD degree in 2005 and was soon employed as an Assistant Professor at the Department of Computer Science, University of Copenhagen (DIKU). He has served as chairman of the OpenTissue open source project since 2007, and was one of the people who started the project in late 2001. In 2008, he received a NVIDIA professor partnership and became a member of the organizing committee for the Virtual Reality and Interactive Physical Simulation (VRIPHYS) conference. Erleben was employed as an Associate Professor in 2009 and appointed head of the eScience Research School at the Faculty of Science in 2010. Currently he serves as Deputy Head of Department of Education at DIKU. Erleben works with interactive computer simulation and has particular interest in computational contact mechanics and computational fluid dynamics.

Adam Bargteil is an assistant professor in the School of Computing at the University of Utah. He completed his PhD degree in computer science at the University of California at Berkeley, where he worked in the Berkeley Computer Animation & Modeling group. He then spent two years as a postdoctoral fellow working in the Graphics Lab at Carnegie Mellon University. Bargteil's research interests are in computer graphics and animation, especially using physical simulation for computer animation. He is also interested in scientific computing, numerical methods, computational physics and computational geometry.

Jens Fursund received his M.Sc. degree in Digital Media Engineering from the Technical University of Denmark in 2010. He was then employed as a Research and Innovation Scientist at the Alexandra Institute, Denmark. Since October 2012 he has been working as an R&D Engineer at Industrial Light & Magic, CA, USA. His research interests include computer graphics, physics-based simulation and computational geometry.

Brian Bunch Christensen received his PhD in fluid animation for visual effects from the Department of Computer Science at Aarhus University in 2010. His research interests include fluid simulation, real-time rendering, and ray-tracing techniques.

Jakob Andreas Bærentzen received his MSc and PhD degrees from the Technical University of Denmark. He is now an Associate Professor in the Department of Applied Mathematics and Computer Science at the Technical University of Denmark. Andreas Barentzen's research interests are focused on digital 3D shapes. In particular, he investigates shape representation and manipulation methods for applications such as interactive sculpting, simulation and modeling of dynamic phenomena, procedural synthesis of 3D models, and the creation of digital prototypes. He is also interested in many aspects of real-time graphics.

Robert Bridson is an Associate Professor at the University of British Columbia, in the Computer Science Department. He is also an active industrial researcher, best known for helping develop fluid simulation software used in many recent films; his screen credits include *The Hobbit: An Unexpected Journey* and *The Adventures of Tintin: The Secret of the Unicorn*.