DOI: 10.1111/cgf.14098
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2020
J. Bender and T. Popa
(Guest Editors)

Volume 39 (2020), Number 8

Efficient 2D Simulation on Moving 3D Surfaces

D. Morgenroth[1], S.Reinhardt[1,2], D. Weiskopf[2], and B. Eberhardt[1]

[1]Media University Stuttgart, Germany
[2]Visualization Research Center, University of Stuttgart, Germany

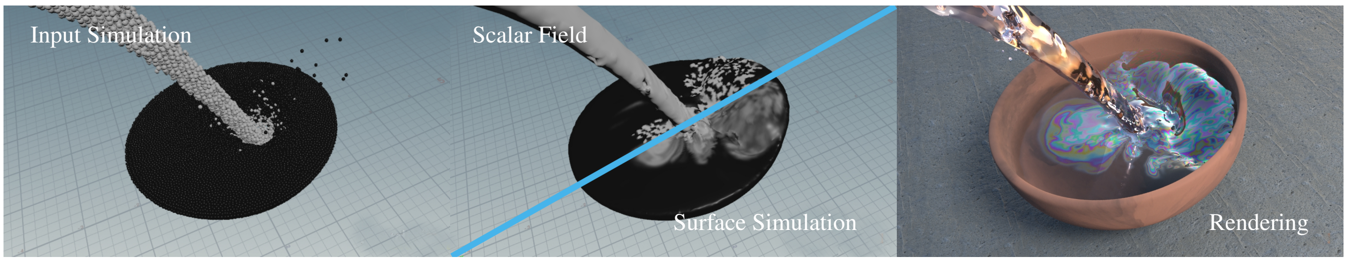


Figure 1: *Water polluted with oil is poured into a cup. We simulate the oil film on top of the existing fluid simulation. The velocity and mass are taken from the existing animation. On the left is the coarse input simulation. In the upper half of the middle image is the resulting scalar field. The lower half shows the result of our method. Our high-resolution 2D simulation adds convincing visual details to the coarse input simulation. The right image displays the final rendering that needs the fine details from our surface simulation to generate the high-resolution thin-film interference effects.*

Abstract
We present a method to simulate fluid flow on evolving surfaces, e.g., an oil film on a water surface. Given an animated surface (e.g., extracted from a particle-based fluid simulation) in three-dimensional space, we add a second simulation on this base animation. In general, we solve a partial differential equation (PDE) on a level set surface obtained from the animated input surface. The properties of the input surface are transferred to a sparse volume data structure that is then used for the simulation. We introduce one-way coupling strategies from input properties to our simulation and we add conservation of mass and momentum to existing methods that solve a PDE in a narrow-band using the Closest Point Method. In this way, we efficiently compute high-resolution 2D simulations on coarse input surfaces. Our approach helps visual effects creators easily integrate a workflow to simulate material flow on evolving surfaces into their existing production pipeline.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

1. Introduction

In typical workflows for generating digital visual effects, a team of VFX artists iteratively refines a given sequence until they achieve good quality. Going from rough storyboards over blocked animation to the final shot with many layers of physical simulations, each shot passes through the VFX pipeline, where domain specialists add new effects and details. With physical simulations, often an effect is finished and approved before secondary effects are layered on top of it. For example, after simulating the fluid flow of a water surface, secondary effects like splashes and foam [TFK*03] [IAAT12] are added on top of this "basic" water simulation, which we will call "base animation". In this context, we propose a method to add secondary effects on top of the base animation by solving a PDE on the surface. As an example, we simulate a thin-film 2D fluid simulation on top of a possibly precomputed, bulk fluid simulation as shown in Fig. 1. We employ a one-way coupling to transfer momentum and mass from the 3D fluid simulation to the surface simulation. We base our coupling on physical derivations and provide them with plausible parameters to control its effects. This coupling allows us to iterate on the secondary effects with a consistent high-resolution 2D simulation on top of the unchanged coarse 3D input simulation.

The goal is to solve a partial differential equation (PDE) on a moving 2-manifold. The approach can be used to address different types of problems that require solving PDEs like fluid flow, reaction-diffusion texture synthesis, or 2D wave equations. Fig. 2

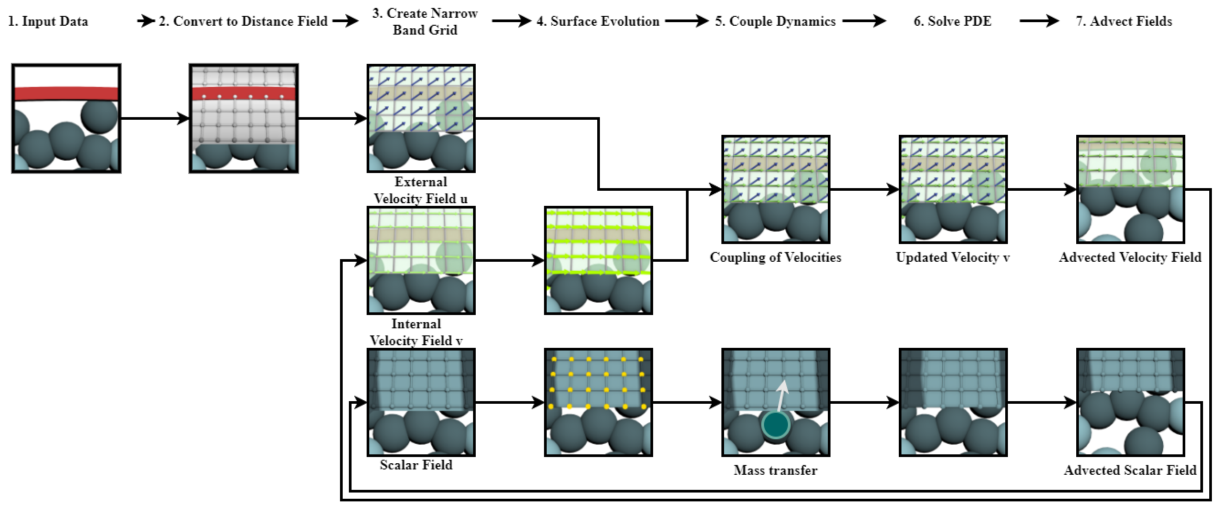


Figure 2: *The seven steps of our method as a flow chart. The steps are placed in order of execution from left to right. After the input of the external data (Step 1), the data is transferred to the narrow-band (Step 2 and 3). Next, the surface is evolved (Step 4) and the outer process is coupled with the inner dynamics (Step 5), before solving the set of PDEs on the surface (Step 6). Finally, the fields are advected (Step 7).*

depicts the steps of our approach. We start with a moving surface as input geometry. After converting the input data into a distance field and transferring values into a narrow-band grid around the surface, we introduce quantities from the input 3D simulation into our surface domain in a coupling step where the strength of the coupling can be driven by parameters. Then, we use the Closest Point Method (CPM) to embed the 2-manifold in the 3D space of the moving surface and solve 2D PDEs in a 3D narrow-band.

With our method, we can model secondary effects very efficiently, and we evaluate our approach by simulating a variety of effects such as pouring an oil film into water, simulating reaction-diffusion on a water surface, or the surfactants in a heated soap bubble. The source code is available on GitHub [MRWE20b].

2. Related Work

Secondary effects can be simulated on top of the base animation in two different ways. The first way is to directly integrate them into the simulation, for example, bubbles in foam [TSS*07], [CPPK07], or [IBAT11]. Here, the secondary effects are two-way coupled with the main fluid. Modeling the secondary effects in such a way does not allow for post-processing of existing simulations and animations without the need to re-simulate. To address this problem, we decouple the surface simulation from the input simulation. Furthermore, this decoupling allows us to use a higher resolution on the surface simulation and add fine details on top of coarse inputs.

The second way to add additional effects like splashes, bubbles, and foam is to simulate them with the main fluid but not to exert forces back onto the main simulation, e.g., [TFK*03], [KLKK12], or [IAAT12]. Splashes and bubbles need true 3D solvers as they extend into, or out of, the surface. Foam only lives on the surface and can be rendered [ADAT13] by tagging surface particles of the original simulation or by moving texture patches with the surface [GDP16]. In contrast, we are interested in effects that are limited to the 2D surface, but we want to simulate a finer resolution on the surface than the original 3D fluid simulation offers.

Solving PDEs on 2-manifolds has been addressed earlier. Some solutions are designed for special cases of geometry, e.g., spheres [HH16]. Instead, we are interested in a generic solution. Stam [Sta03] simulated fluid flows on static Catmull-Clark surfaces or Shi et al. [SY04] and Vantzos et al. [AVW*15] on static triangle meshes. Azencot et al. [AWO*14] model fluids on triangle surfaces using their vorticity by a time-varying scalar function. Ruuth et al. [RM08] proposed the Closest Point Method (CPM) to calculate PDEs on surfaces. This was applied by Macdonald et al. [MMR13] to calculate reaction-diffusion terms on surfaces derived from point clouds. CPM on static surfaces can be calculated in real time [AMT*12]. The theory for solving PDEs on evolving surfaces is explained by Xu and Zhao [XZ03]. Examples are the simulation of soap film on bubbles [ISN*20] where the PDE is solved on a triangle mesh, or computing the flow within a soap bubble on a staggered spherical grid [HIK*20]. On evolving surfaces, Mercier et al. [MBT*15] solved a PDE in a 3D narrow-band grid using the CPM to enhance the original simulation with additional turbulence. Closest to our approach is the Semi-Lagrangian Closest Point Method [AW13]. In comparison to existing CPM-based methods, we add equations for conservation of mass. This allows us to correctly adapt scalar values when the surface area changes. We also adjust vector quantities to account for surface changes. Furthermore, we model a one-way coupling using mass and momentum transfer to create a plausible linkage between the evolution of the input surface and the behavior of the resulting 2D simulation.

3. Modeling PDEs on Evolving Surfaces

In this section, we describe the fundamentals of our method and how we model the physics on the surface. We additionally describe the different physical phenomena that we model as secondary effects on the surface.

3.1. Overview

Consider an evolving surface M. Such a surface can result from a finite-difference or Smoothed Particle Hydrodynamics (SPH) simulation, a keyframe animation, or any time-dependent process. On M, we define another process (e.g., the simulation of a substance) with a set of PDEs and couple it to the base animation M results from. Therefore, the overall system is divided into three aspects:

- **Evolution of the surface M**:
  Due to the surface evolution, the space on which we model the dynamic process is altered. In this step, we account for this fact, i.e., how quantities evolve alongside the surface.
- **Coupling**:
  This aspect determines how the base animation affects the one on the surface.
- **Modeling the dynamics on the surface**:
  This step accounts for the secondary dynamic process and how it is modeled on the surface, i.e., how a set of PDEs can be solved on the surface.

An example of such a process could be a drop of ink on a plastic sheet moving in space. The first aspect (evolution of the surface) describes how the ink is transported in space when the plastic sheet is moving in normal direction. The second aspect (coupling) accounts for the movement of the plastic sheet in tangential direction, i.e., the acceleration of the ink due to friction forces, and the third aspect (the dynamic process on the surface) is concerned with the fluid behavior of the ink itself.

3.2. Evolution of the Surface

To model such a dynamic process on an evolving surface $M \subset \mathbb{R}^3$, we need to define scalar as well as vector-valued quantities on it. To this end, we attach scalar quantities $a(\mathbf{p},t) \in \mathbb{R}$ and vectorial quantities $\mathbf{v} := \mathbf{v}(\mathbf{p},t) \in T_\mathbf{p}M$ to every point $\mathbf{p} \in M$, where $T_\mathbf{p}M$ denotes the tangent space at point $\mathbf{p}$ defined by the surface normal $\mathbf{n} := \mathbf{n}(\mathbf{p},t)$ and t denotes a certain point in time. Typical quantities would be, e.g., mass density and velocity for fluid flow on the surface. The velocity field $\mathbf{u} := \mathbf{u}(\mathbf{p},t)$ defines how the surface evolves, i.e., how $M(t_0 + \Delta t)$ results from $M(t_0)$, where $\Delta t \in \mathbb{R}_{>0}$. For convenience, we will write M instead of $M(t_0)$ and M' for $M(t_0 + \Delta t)$. In the following, the surface evolution characterized by $\mathbf{u}$ will be referred to as the outer process and the one on the surface will be called inner dynamics.

We construct a map O that describes the evolution of quantities due to the outer process. To this end, we define the space $\mathcal{M}$:

$$\mathcal{M} = \bigcup_{\mathbf{p}\in M} \{\mathbf{p}\} \times \mathbb{R} \times T_\mathbf{p}M\,. \tag{1}$$

The quantities needed for the inner dynamics can now be described as elements of $\mathcal{M}$. Without loss of generality, we consider only one scalar and one vectorial quantity attached to point $\mathbf{p}$. It is possible to map an arbitrary number of quantities to $\mathbf{p}$ in the same way. The map O relates elements from $\mathcal{M}$ to elements in space $\mathcal{M}' = \bigcup_{\mathbf{p}'\in M'} \{\mathbf{p}'\} \times \mathbb{R} \times T_{\mathbf{p}'}M'$:

$$O : \mathcal{M} \to \mathcal{M}' : \begin{pmatrix}\mathbf{p}\\ a\\ \mathbf{v}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{p}'\\ a'\\ \mathbf{v}'\end{pmatrix}. \tag{2}$$

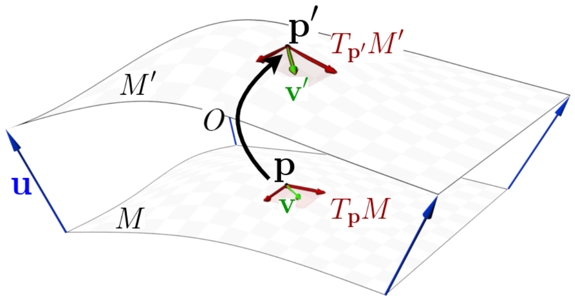


Figure 3: *The map O relates $\mathbf{p} \in M$ and $\mathbf{p}' \in M'$ and also maps tangent space $T_\mathbf{p}M$ into $T_{\mathbf{p}'}M'$.*

An illustration of O can be found in Fig. 3.

We divide the influence of the outer process defined by $\mathbf{u}$ in normal $\mathbf{u}_n$ and tangential $\mathbf{u}_t$ components. The tangential component is used to model friction between the outer and inner dynamics and discussed in Section 3.3. For now, we assume that the dynamics are coupled without friction and, therefore, $\mathbf{u}_t$ does not affect the inner dynamics, i.e., $m \in \mathcal{M}$ is only altered by $\mathbf{u_n}$.

The rate of change $\frac{D_O}{Dt}$ that a point $\mathbf{p}$ experiences from to action of O is then defined by $\tilde{\mathbf{u}}_\mathbf{n} := k\mathbf{u_n}$, where $\frac{D_O}{Dt}$ denotes the material derivative and $k \in \mathbb{R}$ has to be chosen in such a way that $\mathbf{p}' \in M'$ holds. The motion of point $\mathbf{p}$ is directly governed by $\frac{D_O\mathbf{p}}{Dt}$, in particular its velocity is $\tilde{\mathbf{u}}_\mathbf{n}$ and, hence, $\frac{D_O\mathbf{p}}{Dt} = \tilde{\mathbf{u}}_\mathbf{n}$.

A scalar quantity a is advected alongside the point $\mathbf{p}$. We distinguish between two types of scalar quantities: intensive and extensive ones [Red70]. An extensive property is a global property (e.g., mass or volume). Such a property is only advected alongside $\mathbf{p}$ and not changed, i.e., $\frac{D_O a}{Dt} = 0$. In contrast, if a describes an intensive physical property, such as density, we demand that

$$0 = \frac{D_O}{Dt} \int_{M(t)} a(\mathbf{p},t)\, dM(\mathbf{p},t) \tag{3}$$

holds true, where $dM(\mathbf{p},t)$ are infinitesimal surface elements at position $\mathbf{p}$ on the surface M at time t. In the following, we will skip t and $\mathbf{p}$ for convenience. Eq. 3 ensures that the total amount of such a quantity does not change if no phenomena like mass transfer or chemical reactions are present. In this case, the rate of change of an intensive quantity a is given by

$$\frac{D_O a}{Dt} = -a(\nabla \cdot (\tilde{\mathbf{u}}_\mathbf{n})_{T_\mathbf{p}M}), \tag{4}$$

where $\mathbf{u}_{T_\mathbf{p}M} = (I - \mathbf{n}\mathbf{n}^T)\mathbf{u}$ is the velocity projected onto the tangent space $T_\mathbf{p}M$ at point $\mathbf{p}$. This means that, if the surface diverges, the concentration of a decreases, and vice versa. A detailed derivation is provided in Appendix A.

When advecting a vectorial quantity $\mathbf{v}$, we demand that $\mathbf{v}' \in T_{\mathbf{p}'}M'$ holds after applying O. This can be ensured by considering $\mathbf{v}$ to be a small linear material line element subjected to the velocity field $\tilde{\mathbf{u}}_\mathbf{n}$. Its rate of change is the difference of the velocities at the two ends of the element and can be described by $\frac{D_O\mathbf{v}}{Dt} = \nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}$. In our model, the vector field $\mathbf{v}$ describes the velocity of the inner dynamics. As the directions of the velocities are constrained by the

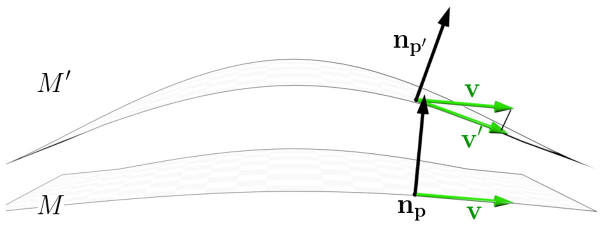


Figure 4: *The map O transforms velocity vector $\mathbf{v}$ to vector $\mathbf{v}'$. First, $\mathbf{v}$ is advected to the point $\mathbf{p}'$. Next, it is projected back into $T_{\mathbf{p}'}M'$ and scaled to conserve momentum.*

surface evolution, the total momentum will be altered and, therefore, cannot be conserved. We can only ensure that the sum of the absolute values does not change, i.e., we can enforce that

$$0 = \frac{D_O}{Dt} \int_M \|a\mathbf{v}\| \, dM \tag{5}$$

will hold. Eq. 5 leads to

$$\frac{D_O \|\mathbf{v}\|}{Dt} = 0. \tag{6}$$

Hence, we need to adjust $\frac{D_O \mathbf{v}}{Dt}$, so that the length of $\mathbf{v}$ is conserved. Using first-order Taylor expansion, the change of the length can be computed by $\mathbf{v} \cdot \nabla \tilde{\mathbf{u}}_{\mathbf{n}} \mathbf{v} \frac{\mathbf{v}}{\|\mathbf{v}\|^2}$. A detailed derivation can be found in Appendix B. This term is then subtracted to adhere Eq. 6. Metaphorically, this means that $\mathbf{v}$ is advected and rotated back onto the surface. An illustration of the advection of a vectorial quantity is given in Fig. 4.

If a is an intensive property, the combined material derivative of the map O then reads as

$$\frac{D_O}{Dt} O = \begin{pmatrix} \tilde{\mathbf{u}}_{\mathbf{n}} \\ -a(\nabla \cdot (\tilde{\mathbf{u}}_{\mathbf{n}})_{T_{\mathbf{p}}M}) \\ \nabla \tilde{\mathbf{u}}_{\mathbf{n}} \mathbf{v} - \mathbf{v} \cdot \nabla \tilde{\mathbf{u}}_{\mathbf{n}} \mathbf{v} \frac{\mathbf{v}}{\|\mathbf{v}\|^2} \end{pmatrix}. \tag{7}$$

Note that $\frac{D_O}{Dt} a$ denotes the rate of change of a quantity a induced by the operator O and $\frac{D_O}{Dt} O$ denotes the combined material derivative of O.

3.3. Coupling

So far we assumed that the outer process influences the inner dynamics solely in normal direction. Next, we discuss how the tangential part $\mathbf{u}_t = \mathbf{u} - \mathbf{u}_n$ will influence the inner dynamics. This implies that friction-less coupling is modeled, i.e., matter slides on the surface and no adhesion is present. We model adhesive forces to reduce the relative velocities $\mathbf{v}^{\text{rel}} = \mathbf{v} - \mathbf{u}_t$ between the outer process and inner dynamics as

$$\frac{D_c \mathbf{v}}{Dt} = -s_1 \mathbf{v}^{\text{rel}}, \tag{8}$$

where s_1 is a user-defined coefficient and $\frac{D_c}{Dt}$ denotes the rate of change induced by coupling effects. Similar to Section 3.2, we could also consider $\mathbf{v}$ to be a small linear material line element exposed to the velocity $\mathbf{u}_t$ and its rate of change would then read as $\left(\nabla_{T_{\mathbf{p}}M} \mathbf{u}_t\right) \mathbf{v}$, where $\nabla_{T_{\mathbf{p}}M}$ is defined as $\nabla_{T_{\mathbf{p}}M} = (I - \mathbf{n}\mathbf{n}^T)\nabla$. We combine both views and model the total rate of change of $\mathbf{v}$ caused by coupling effects as

$$\frac{D_c \mathbf{v}}{Dt} = -s_1 \mathbf{v}^{\text{rel}} + s_2 \left(\nabla_{T_{\mathbf{p}}M} \mathbf{u}_t\right) \mathbf{v}. \tag{9}$$

The outer process induces sinks and sources to the inner dynamics modeled on the surface. Therefore, we model transfer of intensive physical quantities (e.g., density) from the outer process to inner dynamics via sinks and sources on the surface. Inspired by Fick's laws of diffusion, we want that the concentration a^{out} from the outer process and the concentration a of the inner dynamics will align. In other words, the concentration of a substance on the surface will be changed by

$$\frac{D_c a}{Dt} = -s_3 a^{\text{rel}}, \tag{10}$$

to reduce the concentration difference $a^{\text{rel}} = a - a^{\text{out}}$. The speed of the reduction is determined by the factor s_3. If only sources are modeled, the concentration difference is restricted to negative values, i.e., $a^{\text{rel}} \leq 0$. Sinks are modeled by restricting $a^{\text{rel}} \geq 0$.

3.4. Modeling the Dynamics on the Surface

After discussing the surface evolution and coupling, we describe how dynamics on the surface can be modeled. To describe such a process we use a set of PDEs, e.g.,

$$\frac{D_s \mathbf{v}}{Dt} = F_1 \left(t, a, \mathbf{v}, \partial_1 a, \partial_2 a, ..., \partial_1 \mathbf{v}, \partial_2 \mathbf{v}, ...\right) \text{ and} \tag{11}$$
$$\frac{D_s a}{Dt} = F_2 \left(t, a, \mathbf{v}, \partial_1 a, \partial_2 a, ..., \partial_1 \mathbf{v}, \partial_2 \mathbf{v}, ...\right), \tag{12}$$

where the functions F_1 and F_2 characterize the phenomena modeled on the surface. The term $\partial_i := \left\{ \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3}} : |\alpha| = i \right\}$ denotes the set of all partial derivatives of the order i, with α being a multi-index. For example, if fluid flow is modeled, F_1 describes the momentum conservation and F_2 the mass conservation of the Navier-Stokes equations.

To solve Eqs. 11 and 12 on the surface, we only allow for tangential deviations. To this end, we solve them locally in the corresponding tangent spaces $T_{\mathbf{p}}M$. This implies that, instead of using the ordinary spatial derivations, we project them onto the surface, e.g., the operator ∇ is replaced by $\nabla_{T_{\mathbf{p}}M}$. The governing equations for our model are then given by:

$$\frac{D\mathbf{v}}{Dt} = \frac{D_O \mathbf{v}}{Dt} + \frac{D_c \mathbf{v}}{Dt} + \frac{D_s \mathbf{v}}{Dt} \text{ and} \tag{13}$$
$$\frac{Da}{Dt} = \frac{D_O a}{Dt} + \frac{D_c a}{Dt} + \frac{D_s a}{Dt}. \tag{14}$$

While the outer process is ignored in Equations 11 and 12, it is included in Equations 13 and 14, i.e., they govern the overall dynamics on the surface.

Next, we describe different physical phenomena that we use in this paper to show the versatility of our model.

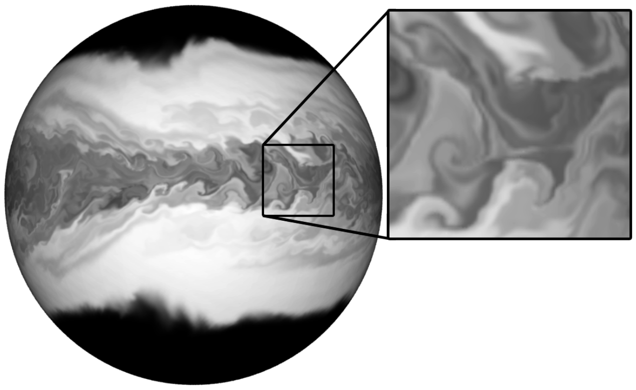


Figure 5: *A rotating sphere where we couple the velocity of the internal simulation with the velocity of the sphere. Details are added with artificial vorticity and coupling noise. Simulation resolution is 207 × 207 × 207.*

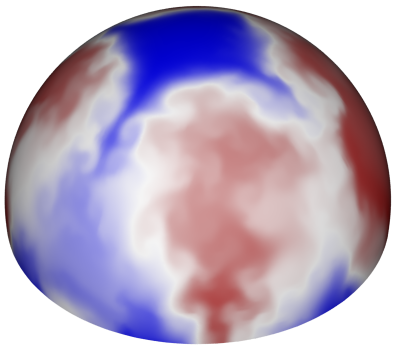


Figure 6: *Thermal convection on a hemisphere. The temperature is color-coded, where the temperature rises from blue over white to red. Some areas are heated and others are cooled. Due to these temperature differences, buoyancy-driven flow arises. Simulation resolution is 207 × 107 × 207.*

3.5. Examples

We model different physical phenomena on the surface and, therefore, employ different sets of equations.

Fluid Flow. One application of our model is the simulation of fluid flow on an evolving surface. Figs. 5, 7, and 8b show examples of such a flow. The surface rotates and, due to friction forces, the fluid on the surface starts to move. To model viscous fluid flow on the surface we use the Navier-Stokes momentum equation:

$$\frac{D_s\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla_{T_pM}P + \nu\nabla^2_{T_pM}\mathbf{v} + \frac{1}{\rho}\mathbf{F}_b\,, \tag{15}$$

where $\mathbf{v}$ is the fluid's velocity, ν the viscosity constant, P is the pressure, and $\mathbf{F}_b$ are body forces (e.g., gravity).

The evolution of density ρ is modeled by using the general continuity equation:

$$\frac{D_s\rho}{Dt} = \sigma - \rho(\nabla_{T_pM}\cdot\mathbf{v})\,, \tag{16}$$

where σ is the rate of generation of the substance per unit volume, i.e., it is used to model sinks ($\sigma < 0$) or sources ($\sigma > 0$). We model incompressible fluid flow on the surface and only allow for density changes due to surface divergence, i.e., we set $F_2(\rho,\mathbf{v}) = 0$. As a result, Eq. 16 simplifies to a volume conservation law:

$$\nabla_{T_pM}\cdot\mathbf{v} = \frac{\sigma}{\rho}\,. \tag{17}$$

If no sinks or sources are present (i.e., $s_3 = 0$), Eq. 16 simplifies to $\nabla_{T_pM}\cdot\mathbf{v} = 0$. Note that if the outer process is divergence-free, i.e., $\nabla\cdot\mathbf{u} = 0$, we obtain an incompressible flow because $\frac{D\rho}{Dt} = 0$.

Buoyancy-induced Flow. Fig. 6 shows a static hemisphere. Some areas on the surface are heated, others cooled by outer temperature constraints. We model temperature transfer according to Eq. 10 and, therefore, the fluid on the surface changes its temperature. Natural convection arises due to temperature differences in the fluid.

Such buoyancy-driven flow can be modeled using the so-called Boussinesq approximation [Bou97]. The Boussinesq approximation assumes incompressible flow and that variations of density only occur due to temperature differences: $\rho = \rho_0 - \beta\rho_0(\mathcal{T} - \mathcal{T}_0)$, where β is the coefficient of thermal expansion, $\mathcal{T}_0$ the reference temperature, and ρ_0 the reference density. We assume an ideal gas and, therefore, $\beta = \frac{1}{\mathcal{T}_0}$. If we assume gravity as the only body-force present, Eq. 15 becomes

$$\frac{D_s\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla_{T_pM}P + \nu\nabla^2_{T_pM}\mathbf{v} + \left(1 - \frac{\mathcal{T}}{\mathcal{T}_0}\right)\mathbf{g}\,, \tag{18}$$

with $\mathbf{g}$ being the gravitational constant. To model changes in the temperature field we use the convection-diffusion equation:

$$\frac{D_s\mathcal{T}}{Dt} = \mu_d\nabla^2_{T_pM}\mathcal{T}\,, \tag{19}$$

where we assume a constant diffusion coefficient μ_d.

Reaction-diffusion. Reaction-diffusion is commonly used to model chemical reactions of one or more substances (e.g., a substance is transformed into another due to chemical reactions) and the diffusion of the substance(s) in space. Fig. 8a shows such a process. Two chemical substances diffuse and react with each other. In this example, we model a two-component reaction-diffusion process given by

$$\frac{D_sf_1}{Dt} = R_1(f_1,f_2) + \mu_{d_1}\nabla^2_{T_pM}f_1 \text{ and} \tag{20}$$

$$\frac{D_sf_2}{Dt} = R_2(f_1,f_2) + \mu_{d_2}\nabla^2_{T_pM}f_2\,. \tag{21}$$

The functions R_1 and R_2 are the reaction terms and characterize the system.

We use the Gray Scott Model [GS84] to define the reaction terms, i.e., to model how different morphogens react:

$$R_1(f_1,f_2) = -f_1f_2^2 + \beta(1-f_1)\,, \tag{22}$$

$$R_2(f_1,f_2) = f_1f_2^2 - (\beta+\gamma)f_2\,, \tag{23}$$

To generate the example illustrated in Fig. 8a, we set the parameters to $\beta = 0.03$, $\gamma = 0.06$, $\mu_{d_1} = 0.1$, and $\mu_{d_2} = 0.1$.

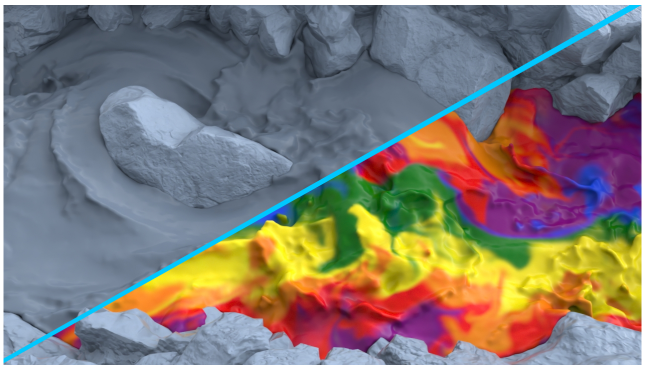


Figure 7: *Fluid flow simulation on top of a FLIP fluid riverbed simulation. Performing a surface simulation of a color-band demonstrates that we can add fine-scaled details. The split screen shows the input fluid simulation on the left and the added surface flow on the right.*

4. Connecting to the Base Animation

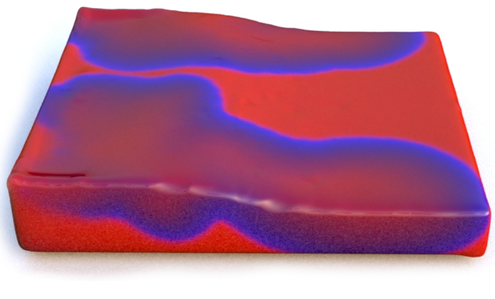
In this section, we describe how we apply the model from Section 3 to create a simulation system. Our method can be divided into seven steps, as illustrated in Fig. 2. We start with a coarse input (Fig. 2, Step 1), e.g., a fluid simulation. Then, we create a signed distance field from this simulation (Fig. 2, Step 2) and write the simulation properties such as velocity, density, or color into 3D fields that are laid out in a narrow-band grid, which is our generic input format (Fig. 2, Step 3). We bring quantities from the outer process into our surface domain in a coupling step and compensate for the surface evolution as described in Section 4.4 (Fig. 2, Step 4 and 5). Then, we simulate 2D details to enhance the coarse input by solving a PDE in the 2D surface space (Fig. 2, Step 6) that results in a velocity field that advects all grids in the last step (Fig. 2, Step 7).

To solve a PDE on a surface we use the Closest Point Method (CPM) [RM08] calculated in a 3D narrow-band. The main idea of the CPM is that if the values of a field are constant along the normal of the surface, many equations calculated in this 3D field are the same as if they were calculated in the tangent spaces of the surface. Section 4.3 describes the process of converting 3D fields into fields that have constant values along the surface normal.

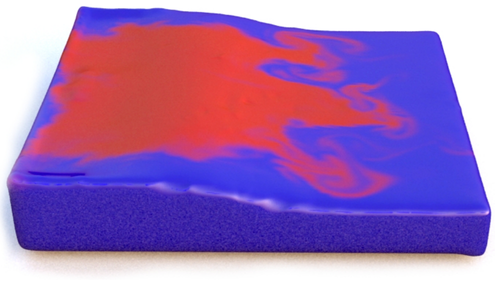
We implemented our method where each part is interchangeable using a system of modules that change data flowing through the system. In the following, we will describe each step in detail.

4.1. Data Input

The first step implements data input and is able to process a variety of base animation types. In our examples, we consider keyframe animation (Fig. 5), SPH particle simulation (Fig. 12 and 8a), and fluid implicit particle (FLIP) simulation (Fig. 7). The only requirement for an input is that we can convert the surface geometry data into a signed distance field and that values for the surface velocity can be generated on the surface.



(a) *Reaction-diffusion*



(b) *Fluid flow*

Figure 8: *With our approach, we can model different behaviors on the same input data. The upper image shows a reaction-diffusion simulation on top of a dam break SPH simulation. The lower image shows a fluid flow on top of the same input simulation. Simulation resolution is 199 $\times$ 50 $\times$ 169.*

4.2. Convert to Signed Distance Field

For converting polygon meshes to signed distance fields, several methods are available, e.g., [SGGM06], [XB14], or [KDBB17]. A special case arises when we convert particle systems to distance fields. Inside the fluid, there are particles everywhere. Here, the distance field obtained by these methods is not usable since we need the distance to the surface and not to the nearest particle. For particle-based simulations, we can either transform the simulation to a polygon mesh first or go directly from particles to signed distance fields by defining an implicit surface from the particles [ZB05] and then take the distance to this implicit surface.

4.3. Create Narrow-Band Grid

We create a narrow-band around the surfaces to capture the velocity, density, and other properties of the flow from the nearest surface point. We only fill cells near the surface based on the distance field. Cells that a farther away than a certain threshold are left empty, as illustrated in Fig. 9. We keep track of two velocity fields, the internal velocity field (denoted as **v** in Section 3) and the external velocity field (denoted as **u**). We write the velocity of the incoming surface into the external velocity grid. Depending on the use case, we either initialize the internal velocity grid with a snapshot of the external velocity (for example, Fig. 12), or we create a custom initialization routine to define the initial internal velocity, for example, by initializing with vanishing velocity (Fig. 5). We also keep track of the scalar properties that we use in our simulation, e.g., the concentration of substances for the reaction-diffusion example.

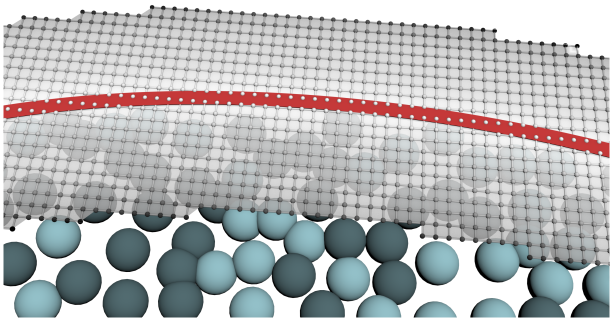


Figure 9: *The distance field is stored in a narrow-band around the surface. In the same way, velocity and scalar fields are stored in a sparse data structure.*

To use the CPM our inputs must be narrow-band grids where the values are constant along the normal direction of the surface. Using an integer look-up grid that stores the cell coordinates of the closest point of the surface in each grid cell, we can fill in a grid with values from the closest surface point. We refer to this step as the CPM extension. Here, we extend the values near the surface along the normal direction. The CPM extension is our final step in the narrow-band creation phase.

4.4. Surface Evolution

Before we use the velocities or scalar values like color or density in a PDE, we first execute the correction steps for density and velocity, as described in Section 3.

For mass conservation, we have to solve Eq. 4. As a building block for this calculation, we need a module that implements the operator $\nabla \cdot (\tilde{\mathbf{u}}_{\mathbf{n}})_{T_pM}$. This operator will calculate a scalar value for the divergence of the velocity field but with respect to the tangent space of a surface defined by the gradient of the input distance field. The result is then applied to the scalar fields that need correction. The interesting part of this divergence calculation is that we use the original 3D velocities for the divergence operator. We need to take the velocities of adjacent cells into account but projected to the surface normal of the current grid value, not the normals of the respective neighbor cells. An example to better understand the difference is the one of a growing sphere. Although the 2D velocities in surface space are zero at each point, the divergence is not. To get correct 2D divergence where a growing sphere causes sinks, and a shrinking sphere creates sources in the mass conservation equation, we have to project the velocities of neighbor cells using the normal of the current cell.

For the conservation of momentum, the "Project Vector" module alters the velocities based on the surface tangent. Given a source vector grid and a distance field gradient, this module will project the input vectors onto the surface by subtracting the normal vector component from the gradient field. There is an option to maintain the length from the input vector, which resembles rotating the vector down onto the surface, as shown in Fig. 4. To apply Eq. 9, we add a module to calculate the jacobian of a velocity field and use it to change the velocities of a second velocity field. Both modules are applied to the velocity field for the internal velocity.

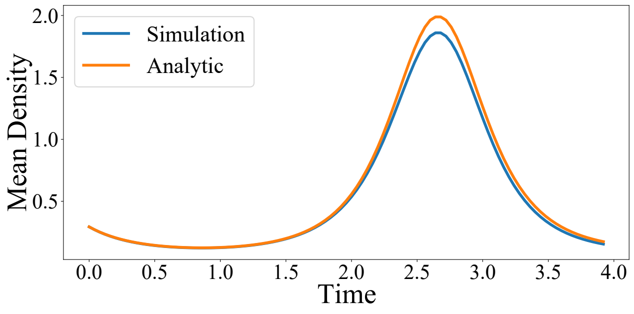


Figure 10: *Simulated density on a growing and shrinking sphere compared to the analytic solution. With our simulation, we can reproduce the analytical result. The numerical diffusion is negligible.*

After the values are adapted based on the surface evolution, we couple them to the values from the underlying simulation.

4.5. Coupling of Dynamics

For coupling the velocities, we send the velocity grids to a module that applies Eq. 9 to the inner velocity. The parameters for $s1$ and $s2$ are exposed to the user. We implemented functionality to add noise, viscosity, and vorticity to the inner velocity. These parameters allow us to improve the coupling with additional details, as can be seen in Fig. 5.

By coupling scalar values from the initial 3D simulation to the 2D space using Eq. 10, we can model mass transfer. The user can choose the parameter $s3$ to drive how much the outer process influences the values in the 2D simulation. A value of 0 would only initialize the 2D simulation, and from then on, the 2D simulation would be independent.

4.6. Solve PDE and Advect

As mentioned, we are operating on 3D grids where values do not change along the normal direction. The CPM allows us to solve PDEs in the surface space using these grids. We implemented different sets of PDEs that can all be solved using our new modules. The modules work in 3D, but we specifically designed them for the fields that result from the CPM extension. Due to the CPM, the gradient naturally operates in surface direction, but the divergence, curl, and Jacobian have to be changed: When applying them, we use the projected version as described in Section 3, i.e., the input vector is projected onto the surface.

To simulate fluid flow on the surface, we solve Eq. 15. Here, we use the projected divergence operator $\nabla \cdot (\tilde{\mathbf{u}}_{\mathbf{n}})_{T_pM}$. The "Divergence-Free" module will remove divergence with respect to the tangent space of a surface and take the divergence of the surface evolution into account. For the buoyancy-induced flow, we scaled the gravity depending on temperature as described in Eq. 18 and added a diffusion module for temperature diffusion. To simulate reaction-diffusion for the example shown in Fig. 8a we wrote a module to implement Equations 20–23. As a last step, we advect all fields using Eq. 13, i.e., with the resulting velocities.

4.7. Implementation

Our modular approach can be easily implemented with existing frameworks like the SideFX™Houdini software package, which is widely used for VFX creation and offers an integration of the OpenVDB libraries [MLJ*13] as a sparse volume representation for the calculations. We separated our algorithm into independent modules and implemented them as a separate operators, to be used in Houdini's simulation node graph framework. The OpenVDB framework offers a data structure, the OpenVDB grid, to create narrow-bands. All subsequent calculations employ this structure and only spend computation time where needed. Our "CPM Extension" module is the central block to build CPM solutions in Houdini. Here, we implemented a nearest-neighbor interpolation as suggested by Kim et al. [KTT13], but also box and quadratic interpolation. To generate the SPH base simulations, we used divergence-free SPH [BK15] with consistent Shepard interpolation [RKEW19]. For more details about SPH, we refer the reader to the SPH tutorial by Koschier et al. [KBST19]. When simulating fluid flow, we used vorticity confinement [FSJ01].

5. Results

5.1. Versatility and Simulation Quality

To show the versatility of our method, we modeled different physical phenomena as described in Section 3.5, and tested them on a variety of scenarios. We chose base animations with different characteristics, from static (Fig. 6) to hand-animated meshes (Fig. 5), from coarse-scale SPH-based fluid simulations, including sudden changes in topology (Fig. 8), to high-resolution multi-phase SPH simulations (Fig. 12). In all of the above situations, we were able to add a fine-scale secondary simulation on top of these surfaces, revealing fine-scale details as promised.

With our approach, we can add effects onto the surface of the simulation. Simulating a second fluid flow on the surface enables one to add another phenomena on top of an existing simulation. As shown in the oil film example (Fig. 12), the effect of oil spreading on the surface is achieved by simulating a second, fine-scale fluid flow on the surface, which is just added to the base-simulation. As mentioned, the level of detail of the secondary simulation can be chosen independently and increasing the simulation resolution of the underlying SPH simulation would not have the same effect. Instead, we would just have a scalar field with higher resolution, like the one shown in Fig. 12b, but would not have simulated the laws of the secondary flow. Also, due to the modeled mass transfer, the oil particles that emerge from below the surface can contribute to our 2D simulation. This coupling introduces significantly more details than just a plain simulation on the surface itself. The amount of detail added is significant even for low-resolution base animations, and it can be further improved (Fig. 12c to 12d) by increasing the resolution. Fig. 8a presents a reaction-diffusion of two chemicals simulated on top of a coarse dam break simulation. This example illustrates that we can simulate not only a second flow equation on the surface but any kind of desired physical phenomena that can be described by a set of PDEs.

In the buoyancy-induced flow example (Fig. 6), we simulate thermal convection and demonstrate emergence of isolated vortices on a static surface, replicating the physical experimental setup by Seychelles et al. [SABK08]. The setup consists of a static hemisphere on a heating plate giving rise to thermal convection. We were able to create fine detailed vortex structures using the Boussinesq approximation.

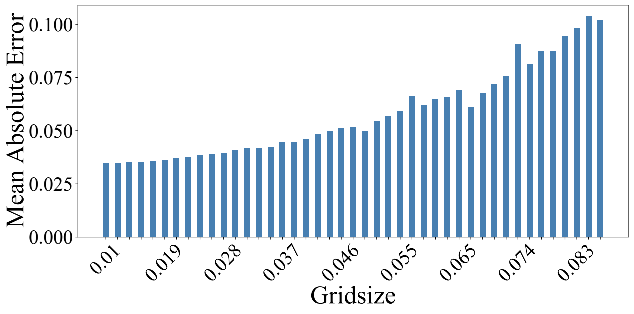


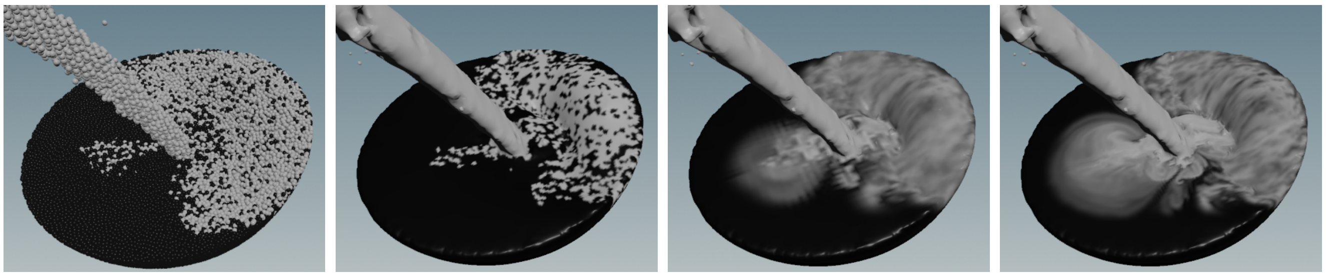
Figure 11: *The mean error for different grid sizes. The smaller the grid cells, the closer the values are to the analytic solution.*

A hand-animated sphere with a periodically oscillating radius was used to test our approach on mass conservation. The mean density per surfactant was calculated and plotted as a function over time as a graph (Fig. 10). In this example, it is possible to analytically calculate the density change that is needed to ensure mass conservation and compare it with our simulated result. As shown in Fig. 10, we can reproduce the analytical solution over time. The small loss in density can be attributed to numerical diffusion and is negligible in this case. To test the accuracy depending on the grid resolution, we simulated this scenario with different grid resolutions. As can be seen in Fig. 11, our mass conservation approach works as expected and the simulation converges to the analytical solution using finer grid resolutions. Further investigations on the convergence order is left for future work.

We refer the reader to the accompanying video to watch some of the examples in motion. Supplementary material discusses the vector length conservation experiment. Moreover, we provide a precompiled version of our plugin for Houdini with two sample scenes for testing on Zenodo [MRWE20a].

5.2. Performance

The performance was measured for every operator individually with the rotating sphere example (Fig. 5). The grid resolution has the most significant impact on the computation time, but also the width of the narrow-band profoundly influences the performance. For the rotating sphere example, the overall computation time per frame with 5 substeps was 28 s. The soap film example from Fig. 12 took approximately 28 s per frame with 2 substeps. The measurements were taken on a notebook with an Intel(R) Core i7-6700K CPU and a NVidia GeForce GTX 1070. The simulation of the underlying fluid is not included in the measurements. As can be seen in Fig. 13, most of the computation time for a single task is spent in the "Divergence Free" operator. This node uses the OpenVDBs Poisson solver with a maximum of 100 iterations to obtain a divergence-free vector field. Second most computation time is



(a) *Source SPH particles* **(b)** *Original scalar field derived from particle attributes* **(c)** *2D simulation with grid resolution of 0.05* **(d)** *2D simulation with grid resolution of 0.02*

Figure 12: *Pouring polluted water into a bowl. The fine-grained simulation enhances the underlying SPH simulation. Our mass transfer even captures oil particles that emerge from under the surface. Different resolutions can be generated independent from the input resolution. This allows iterative refinement of parameters as needed in typical VFX production workflows.*

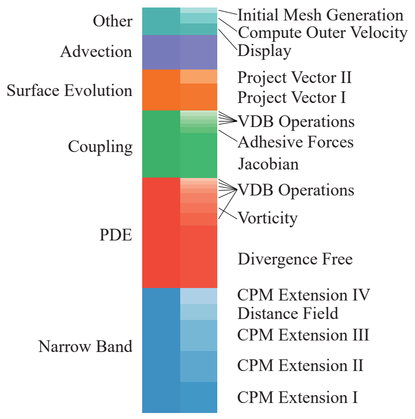


Figure 13: *The calculation costs of our nodes for the CPM methods are of the same magnitude as the existing Houdini operators typically used in simulations.*

spent to compute the Jacobian for the coupling step. In third place is Houdini's advection operator. All other operators have smaller computation costs. This shows that the computation time for our steps is of the magnitude as Houdini's advection node.

6. Conclusion and Future Work

We presented a method to solve PDEs on evolving surfaces where we considered the external movement as the driving process. We derived physically motivated conservation laws and coupling strategies. We coupled simulations of 3D and 2D space in a way that allows us to compute high-resolution 2D simulations on coarse input surfaces efficiently and that exposes physically plausible parameters to control the strength of the coupling. We demonstrated that our approach can be used for different types of problems that require solving PDEs on a surface. We showed how to integrate all necessary steps into a VFX production environment in a modular way so that the building blocks can be reused and we provide a reference implementation as open source. The 2D surface simulation showed to be a usable tool for VFX creation that can add a big visual difference to scenes. By using the CPM there are some limitations to our approach. As most CPM implementations, our method cannot model surfaces with hard edges or high frequencies. These limitations restrict use-cases to smooth surfaces. So far, we assume a surface without boundaries. Future research could integrate different boundary conditions. Testing further types of PDEs to achieve effects like droplet formation or fingering as presented by Vantzos et al. [AVW*15] but on moving surfaces is another topic for future work. Another area of interest is reinjecting 3D fluids from the surface as a simple form of two-way coupling of the simulations, e.g., for water droplets.

Acknowledgements

We want to thank Mariusz Wesierski for his help in setting up the river simulation and his support in all Houdini questions. This work is partly supported by "Kooperatives Promotionskolleg Digital Media" at Stuttgart Media University and the University of Stuttgart.

Open access funding enabled and organized by Projekt DEAL. [Correction added on 26 February 2021, after first online publication: Projekt Deal funding statement has been added.]

Appendix A: Conservation of Intensive Scalar Quantities

In Section 3.2, we stated that Eq. 4 directly follows from Eq. 3 if a represents an intensive scalar quantity. In the following, we provide a detailed derivation of this fact. We want to ensure that the total amount of a does not change if there are no phenomena like mass transfer or chemical reactions present. As the surface evolves, the surface density a needs to be adjusted to ensure that the total amount stays constant. Changing the differentiation and integration order in Eq. 3 results in

$$0 = \int_M \frac{D_O}{Dt}(a\,dM) = \int_M \left(\frac{D_O a}{Dt} dM + a \frac{D_O dM}{Dt} \right). \qquad (24)$$

Since dM evolves, we need to evaluate the term $\frac{D_O dM}{Dt}$.

According to Stone [Sto90], the material derivative of a surface element can be described by

$$\frac{D}{Dt} dM = (\nabla_{T_p M} \cdot \tilde{\mathbf{u}}_n) dM\,, \qquad (25)$$

where $\nabla_{T_\mathbf{p}M} = (I - \mathbf{n}\mathbf{n}^T)\nabla$ is the gradient operator projected onto the tangential space $T_\mathbf{p}M$ at point $\mathbf{p}$, and $\tilde{\mathbf{u}}_n$ is the velocity evolving the surface. Without loss of generality, we assume that the outer process is described by $\mathbf{u}$ instead of $\tilde{\mathbf{u}}_n$ and use $\frac{D\,dM}{Dt}$ instead of $\frac{D_O\,dM}{Dt}$ to derive the rate of change of dM. To describe Eq. 25 in more detail, we first show that the identity

$$\frac{D}{Dt}d\mathbf{M} = (\nabla\cdot\mathbf{u})d\mathbf{M} - (\nabla\mathbf{u})^T d\mathbf{M} \tag{26}$$

holds for an arbitrary oriented surface element $d\mathbf{M} = \mathbf{n}dM$ [Bat00]. To this end, we first investigate how volume elements dV and line elements $d\mathbf{l}$ are changed due to the base animation. We consider the elements to be that small, that they are subject only to pure straining motion and rigid rotations [Bat00].

The rate of change of the volume element dV is described by

$$\frac{D\,dV}{Dt} = \nabla\cdot\mathbf{u}\,dV\,. \tag{27}$$

We assume that the line element $d\mathbf{l}$ is linear (and, therefore, can be described by a vector) and stays approximately straight. Under these assumptions, its rate of change is simply the difference of the velocities at the two ends of the element and can be described by

$$\frac{D\,d\mathbf{l}}{Dt} = \nabla\mathbf{u}\,d\mathbf{l}\,. \tag{28}$$

The volume element dV can also be described by $dV = d\mathbf{M}\cdot d\mathbf{l}$. Inserting dV into Eq. 27 results in:

$$\begin{aligned}(\nabla\cdot\mathbf{u})\,d\mathbf{M}\cdot d\mathbf{l} = \nabla\cdot\mathbf{u}\,dV &= \frac{D(d\mathbf{M}\cdot d\mathbf{l})}{Dt}\\ &= d\mathbf{M}\cdot\frac{D\,d\mathbf{l}}{Dt} + \frac{D\,d\mathbf{M}}{Dt}\cdot d\mathbf{l}\\ &= d\mathbf{M}\cdot(\nabla\mathbf{u}\,d\mathbf{l}) + \frac{D\,d\mathbf{M}}{Dt}\cdot d\mathbf{l}\\ &= ((\nabla\mathbf{u})^T d\mathbf{M})\cdot d\mathbf{l} + \frac{D\,d\mathbf{M}}{Dt}\cdot d\mathbf{l}\,.\end{aligned} \tag{29}$$

As Eq. 29 has to hold for arbitrary line elements $d\mathbf{l}$, we obtain Eq. 26. To get the formulation in Eq. 25 we take the inner-product of $\mathbf{n}$ with Eq. 26 and use the identity $\mathbf{n}^T(\nabla\mathbf{u})^T\mathbf{n} = (\mathbf{n}\mathbf{n}^T\nabla)\cdot\mathbf{u}$:

$$\begin{aligned}\frac{D}{Dt}dM = \frac{D}{Dt}\mathbf{n}^T\mathbf{n}\,dM &= \mathbf{n}^T\frac{D}{Dt}d\mathbf{M}\\ &= \mathbf{n}^T(\nabla\cdot\mathbf{u})\,d\mathbf{M} - \mathbf{n}^T(\nabla\mathbf{u})^T d\mathbf{M}\\ &= (\nabla\cdot\mathbf{u})\,dM - (\mathbf{n}\mathbf{n}^T\nabla)\cdot\mathbf{u}\,dM\\ &= ((I-\mathbf{n}\mathbf{n}^T)\nabla)\cdot\mathbf{u}\,dM = (\nabla_{T_\mathbf{p}M}\cdot\mathbf{u})\,dM\,.\end{aligned} \tag{30}$$

Inserting Eq. 25 into Eq. 24 results in

$$\begin{aligned}0 &= \int_{M(t)}\left(\frac{D_O a}{Dt}dM + a(\nabla_{T_\mathbf{p}M}\cdot\tilde{\mathbf{u}}_\mathbf{n})dM\right)\\ &= \int_{M(t)}\left(\frac{D_O a}{Dt} + a(\nabla_{T_\mathbf{p}M}\cdot\tilde{\mathbf{u}}_\mathbf{n})\right)dM.\end{aligned} \tag{31}$$

As Eq. 31 holds for an arbitrary surface M, we get

$$0 = \frac{D_O a}{Dt} + a(\nabla_{T_\mathbf{p}M}\cdot\tilde{\mathbf{u}}_\mathbf{n}) = \frac{D_O a}{Dt} + a(\nabla\cdot(\tilde{\mathbf{u}}_\mathbf{n})_{T_\mathbf{p}M}). \tag{32}$$

Therefore, the rate of change of a intensive quantity a is defined by

$$\frac{D_O a}{Dt} = -a(\nabla\cdot(\tilde{\mathbf{u}}_\mathbf{n})_{T_\mathbf{p}M}). \tag{33}$$

Appendix B: Length-preserving Evolution of a Vector Field

In Section 3.2, we stated that Eq. 5 is satisfied when setting

$$\frac{D_O\mathbf{v}}{Dt} = \nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \mathbf{v}\cdot\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\frac{\mathbf{v}}{\|\mathbf{v}\|^2}. \tag{34}$$

To confirm that Eq. 34 results from Eq. 5, we first show that $\frac{D_O\|\mathbf{v}\|}{Dt} = 0$ follows from Eq. 5. To this end, we reformulate the right-hand side of Eq. 5 using Eq. 4 and Eq. 25:

$$\begin{aligned}0 = \frac{D_O}{Dt}\int_M \|a\mathbf{v}\|\,dM &= \frac{D_O}{Dt}\int_M a\,\|\mathbf{v}\|\,dM\\ &= \int_M\left(\frac{D_O a}{Dt}\|\mathbf{v}\|\,dM + a\frac{D_O\|\mathbf{v}\|}{Dt}dM + a\,\|\mathbf{v}\|\,\frac{D_O dM}{Dt}\right)\\ &= \int_M a\frac{D_O\|\mathbf{v}\|}{Dt}dM.\end{aligned} \tag{35}$$

This is true for an arbitrary surface M and we obtain:

$$\frac{D_O\|\mathbf{v}\|}{Dt} = 0. \tag{36}$$

To achieve this, we evolve $\mathbf{v}$ with $\nabla\tilde{\mathbf{u}}_\mathbf{n}$ and compensate length changes. The change in length of $\mathbf{v}$ can be expressed by

$$\frac{\mathbf{v}\cdot\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}}{\|\mathbf{v}\|}\frac{\mathbf{v}}{\|\mathbf{v}\|}, \tag{37}$$

and we obtain:

$$\frac{D_O\mathbf{v}}{Dt} = \nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - (\mathbf{v}\cdot\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v})\frac{\mathbf{v}}{\|\mathbf{v}\|^2}. \tag{38}$$

This can be shown when writing the derivative as the limit of the difference quotients. To this end, we use the first-order Taylor expansion of $\mathbf{v}$ with respect to t to define $\hat{\mathbf{v}} = \mathbf{v} + \Delta t\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}$. Compensating for length changes, $\mathbf{v}' := \mathbf{v}(t+\Delta t)$ can be approximated by:

$$\mathbf{v}' = \hat{\mathbf{v}} - (\|\hat{\mathbf{v}}\| - \|\mathbf{v}\|)\frac{\hat{\mathbf{v}}}{\|\hat{\mathbf{v}}\|} \tag{39}$$

and the derivative of $\mathbf{v}$ is written as:

$$\begin{aligned}\frac{D_O\mathbf{v}}{Dt} &= \lim_{\Delta t\to 0}\frac{\mathbf{v}'-\mathbf{v}}{\Delta t} = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left(\Delta t\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - (\|\hat{\mathbf{v}}\| - \|\mathbf{v}\|)\frac{\hat{\mathbf{v}}}{\|\hat{\mathbf{v}}\|}\right)\\ &= \lim_{\Delta t\to 0}\left(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \frac{1}{\Delta t}\left(1-\frac{\|\mathbf{v}\|}{\|\hat{\mathbf{v}}\|}\right)\mathbf{v} + \left(1-\frac{\|\mathbf{v}\|}{\|\hat{\mathbf{v}}\|}\right)\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\right)\\ &= \lim_{\Delta t\to 0}\left(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \frac{\|\hat{\mathbf{v}}\|^2 - \|\mathbf{v}\|^2}{\Delta t\|\hat{\mathbf{v}}\|(\|\hat{\mathbf{v}}\| + \|\mathbf{v}\|)}\mathbf{v} + \right.\\ &\qquad\left.\left(1-\frac{\|\mathbf{v}\|}{\|\hat{\mathbf{v}}\|}\right)\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\right)\\ &= \lim_{\Delta t\to 0}\left(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \frac{2\Delta t\mathbf{v}\cdot(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}) + \Delta t^2\|\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\|^2}{\Delta t\|\hat{\mathbf{v}}\|(\|\hat{\mathbf{v}}\| + \|\mathbf{v}\|)}\mathbf{v} + \right.\\ &\qquad\left.\left(1-\frac{\|\mathbf{v}\|}{\|\hat{\mathbf{v}}\|}\right)\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\right)\\ &= \lim_{\Delta t\to 0}\left(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \frac{2\mathbf{v}\cdot(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}) + \Delta t\|\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\|^2}{\|\hat{\mathbf{v}}\|(\|\hat{\mathbf{v}}\| + \|\mathbf{v}\|)}\mathbf{v} + \right.\\ &\qquad\left.\left(1-\frac{\|\mathbf{v}\|}{\|\hat{\mathbf{v}}\|}\right)\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v}\right) = \nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v} - \frac{\mathbf{v}\cdot(\nabla\tilde{\mathbf{u}}_\mathbf{n}\mathbf{v})}{\|\mathbf{v}\|^2}\mathbf{v}\end{aligned} \tag{40}$$

References

[ADAT13] Akinci N., Dippel A., Akinci G., Teschner M.: Screen space foam rendering. *Journal of WSCG 21*, 3 (2013), 173–182. 2

[AMT*12] Auer S., Macdonald C., Treib M., Schneider J., Westermann R.: Real-time fluid effects on surfaces using the closest point method. *Computer Graphics Forum 31*, 6 (2012), 1909–1923. doi:10.1111/j.1467-8659.2012.03071.x. 2

[AVW*15] Azencot O., Vantzos O., Wardetzky M., Rumpf M., Ben-Chen M.: Functional thin films on surfaces. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2015), pp. 137–146. doi:10.1145/2786784.2786793. 2, 9

[AW13] Auer S., Westermann R.: A semi-Lagrangian closest point method for deforming surfaces. *Computer Graphics Forum 32*, 7 (2013), 207–214. doi:10.1111/cgf.12228. 2

[AWO*14] Azencot O., Weissmann S., Ovsjanikov M., Wardetzky M., Ben-Chen M.: Functional fluids on surfaces. *Computer Graphics Forum 33*, 5 (2014), 237–246. doi:10.1111/cgf.12449. 2

[Bat00] Batchelor G. K.: *An Introduction to Fluid Dynamics*, 8th ed. Cambridge Mathematical Library. Cambridge University Press, 2000. doi:10.1017/CBO9780511800955. 10

[BK15] Bender J., Koschier D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 147–155. doi:10.1145/2786784.2786796. 8

[Bou97] Boussinesq J.: *Théorie de l'écoulement tourbillonnant et tumultueux des liquides dans les lits rectilignes a grande section*, 1 ed. Gauthier-Villars, 1897. 5

[CPPK07] Cleary P. W., Pyo S. H., Prakash M., Koo B. K.: Bubbling and frothing liquids. *ACM Transactions on Graphics 26*, 3 (2007), 97:2–97:6. doi:10.1145/1276377.1276499. 2

[FSJ01] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), pp. 15–22. doi:10.1145/383259.383260. 8

[GDP16] Gagnon J., Dagenais F., Paquette E.: Dynamic lapped texture for fluid simulations. *The Visual Computer 32*, 6 (2016), 901–909. doi:10.1007/s00371-016-1262-8. 2

[GS84] Gray P., Scott S.: Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system a + 2b $\rightarrow$ 3b; b $\rightarrow$ c. *Chemical Engineering Science 39*, 6 (1984), 1087–1097. doi:10.1016/0009-2509(84)87017-7. 5

[HH16] Hill D. J., Henderson R. D.: Efficient fluid simulation on the surface of a sphere. *ACM Transactions on Graphics 35*, 2 (2016), 16:1–16:9. doi:10.1145/2879177. 2

[HIK*20] Huang W., Iseringhausen J., Kneiphof T., Qu Z., Jiang C., Hullin M. B.: Chemomechanical simulation of soap film flow on spherical bubbles. *ACM Transactions on Graphics 39*, 4 (2020). doi:10.1145/3386569.3392094. 2

[IAAT12] Ihmsen M., Akinci N., Akinci G., Teschner M.: Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer 28*, 6 (2012), 669–677. doi:10.1007/s00371-012-0697-9. 1, 2

[IBAT11] Ihmsen M., Bader J., Akinci G., Teschner M.: Animation of air bubbles with SPH. In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (2011), pp. 225–234. doi:10.5220/0003322902250234. 2

[ISN*20] Ishida S., Synak P., Narita F., Hachisuka T., Wojtan C.: A model for soap film dynamics with evolving thickness. *ACM Transactions on Graphics 39*, 4 (2020), 31:1–31:11. doi:10.1145/3386569.3392405. 2

[KBST19] Koschier D., Bender J., Solenthaler B., Teschner M.: Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. In *Eurographics 2019 - Tutorials* (2019), The Eurographics Association. doi:10.2312/egt.20191035. 8

[KDBB17] Koschier D., Deul C., Brand M., Bender J.: An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics 23*, 10 (2017), 2208–2221. doi:10.1109/TVCG.2017.2730202. 6

[KLKK12] Kim P.-R., Lee H.-Y., Kim J.-H., Kim C.-H.: Controlling shapes of air bubbles in a multi-phase fluid simulation. *The Visual Computer 28*, 6 (2012), 597–602. doi:10.1007/s00371-012-0696-x. 2

[KTT13] Kim T., Tessendorf J., Thuerey N.: Closest point turbulence for liquid surfaces. *ACM Transactions on Graphics 32*, 2 (2013), 15:1–15:13. doi:10.1145/2451236.2451241. 8

[MBT*15] Mercier O., Beauchemin C., Thuerey N., Kim T., Nowrouzezahrai D.: Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics 34*, 6 (2015), 202:1–202:10. doi:10.1145/2816795.2818115. 2

[MLJ*13] Museth K., Lait J., Johanson J., Budsberg J., Henderson R., Alden M., Cucka P., Hill D., Pearce A.: OpenVDB: An open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses* (2013), pp. 19:1–19:1. doi:10.1145/2504435.2504454. 8

[MMR13] Macdonald C. B., Merriman B., Ruuth S. J.: Simple computation of reaction–diffusion processes on point clouds. *Proceedings of the National Academy of Sciences 110*, 23 (2013), 9209–9214. doi:10.1073/pnas.1221408110. 2

[MRWE20a] Morgenroth D., Reinhardt S., Weiskopf D., Eberhardt B.: Application and sample scenes for efficient 2D simulation on moving 3D surfaces, 2020. doi:10.5281/zenodo.4009208. 8

[MRWE20b] Morgenroth D., Reinhardt S., Weiskopf D., Eberhardt B.: Source code for efficient 2D simulation on moving 3D surfaces. https://github.com/dimo3d/Cappucino, 2020. 2

[Red70] Redlich O.: Intensive and extensive properties. *Journal of Chemical Education 47*, 2 (1970), 154–156. doi:10.1021/ed047p154.2. 3

[RKEW19] Reinhardt S., Krake T., Eberhardt B., Weiskopf D.: Consistent Shepard interpolation for SPH-based fluid animation. *ACM Transaction on Graphics 38*, 6 (2019), 189:1–189:11. doi:10.1145/3355089.3356503. 8

[RM08] Ruuth S. J., Merriman B.: A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics 227*, 3 (2008), 1943–1961. doi:10.1016/j.jcp.2007.10.009. 2, 6

[SABK08] Seychelles F., Amarouchene Y., Bessafi M., Kellay H.: Thermal convection and emergence of isolated vortices in soap bubbles. *Physical Review Letters 100*, 14 (2008), 144501. doi:10.1103/PhysRevLett.100.144501. 8

[SGGM06] Sud A., Govindaraju N., Gayle R., Manocha D.: Interactive 3D distance field computation using linear factorization. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (2006), pp. 117–124. doi:10.1145/1111411.1111432. 6

[Sta03] Stam J.: Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics 22*, 3 (2003), 724–731. doi:10.1145/1201775.882338. 2

[Sto90] Stone H.: A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface. *Physics of Fluids A: Fluid Dynamics 2*, 1 (1990), 111–112. doi:10.1063/1.857686. 9

[SY04] Shi L., Yu Y.: Inviscid and incompressible fluid simulation on triangle meshes. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 173–181. doi:10.1002/cav.19. 2

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Computer Graphics Forum 22*, 3 (2003), 391–400. doi:10.1111/1467-8659.00686. 1, 2

[TSS*07] THUEREY N., SADLO F., SCHIRM S., MÜLLER-FISCHER M., GROSS M.: Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 191–198. doi:10.2312/SCA/SCA07/191-198. 2

[XB14] XU H., BARBIČ J.: Signed distance fields for polygon soup meshes. In *Proceedings of Graphics Interface* (2014), pp. 35–41. 6

[XZ03] XU J.-J., ZHAO H.-K.: An Eulerian formulation for solving partial differential equations along a moving interface. *Journal of Scientific Computing 19*, 1 (2003), 573–594. doi:10.1023/A:1025336916176. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (2005), 965–972. doi:10.1145/1073204.1073298. 6