Real-time Sponge and Fluid Simulation

V. Burkus, A. Kárpáti, G. Klár and L. Szécsi

Budapest University of Technology and Economics

Abstract

In this paper we present an approach to couple PBD simulation of deformable porous objects with SPH. We propose solutions for simulating the absorption and discharge of fluid by the sponge, and the effect of the fluid on sponge behaviour. We maintain the ability of the original approaches to handle interactions with rigid bodies. Our solution, like PBD in general, is less geared towards physical accuracy, but aims for real-time, visually plausible simulation of these systems, appropriate for interactive VR applications and games.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation; Real-time simulation;***

1. Introduction

Simulation of different kinds of fluids, rigid and soft bodies including fabrics and plastic objects have been studied both in predictively accurate and real-time contexts, but some coupling problems remain open and challenging. This paper presents research on simulating an absorbent, porous soft body with fluids. Even though previous publications have addressed similar couplings, real-time sponge-fluid interaction has not been demonstrated. Our contribution is providing such a solution for 3D porous soft bodies, where not only the capillary effect sucking fluid into the sponge is simulated, but also the fluid moving and deforming the sponge based on physical laws. Given our goal of visually plausible real-time simulation, we have chosen Position Based Dynamics (PBD) for sponge simulation, and Smoothed Particle Hydrodynamics (SPH) for the fluid. Using Position Based Fluid simulation [MM13] would be a strong alternative for future research. We also address real-time rendering of the simulated system.

2. Related work

While grid-based **flow simulation** is possible in real-time, it does not scale favorably with extensive simulation space. Dynamically changing boundaries (e.g. the presence of a plastic material phase) requires conversion of boundary data onto the grid, which may either be too computationally demanding, or restrict resolution. In contrast, particle-based approaches, like SPH [XLF*19], represent liquids as particles of constant mass. SPH evaluates physical quantities at given points by applying smoothing kernels on nearby particles, possibly found using GPU proximity search [WK20]. One fast way of visualizing the results of the simulation is rendering particles as metaballs using screen-space acceleration [SI12]. In entertainment contexts, fluid simulation is often required to adapt artistic or fantastic constraints, while maintaining plausible physical behavior. This is possible to achieve using control particles. A recent example is the work of Zhang et al. [ZYWL15].

Position-based methods of **soft body simulation** replace forces with positional constraints. PBD [BMM17] solves this system by iteratively moving nodes to satisfy individual constraints. Strain Based Dynamics (SBD) [MCKM14] introduces new constraints for triangular and tetrahedral elements, allowing for a PBD-style solution of finite element systems, offering robust recovery from strong deformations. For our sponge simulation, we adapt these tetrahedral constraints.

In their work **coupling fluid and soft-body simulation**, Lenaerts et al. [LAD08] analyze the physical laws governing the interaction of fluids and porous materials, and devise an offline method for the simulation of wet sponges. Fei et al. [FBGZ18] achieve real-time performance by limiting flow simulation in the porous medium to two dimensions, simulating wet fabrics. These papers give equations for *buoyancy*, *pore suction*, and *drag* terms affecting both the liquid and solid phases. They both use different representations for the fluid inside and outside the medium, transferring simulated state between them explicitly or using the APIC method. In contrast, our proposal maintains the Lagrangian representation in both domains. Our goal is to animate **3D** porous solids alongside fluids **interactively**, even including rendering times.

Real-time interaction between fluid and cloth is simulated in a recent study [HZ21]. The point of this work is to *avoid* the interpenetration of fluids into the material, making the method more suitable for waterproof cloth than porous fabrics. Aburumman et al. [ANM*20] combine incompressible SPH and both polygonal meshes and elastic bodies simulated by PBD to achieve a two-way coupling. In the 3D case, they use SBD, which is very close to our

DOI: 10.2312/egs.20221038

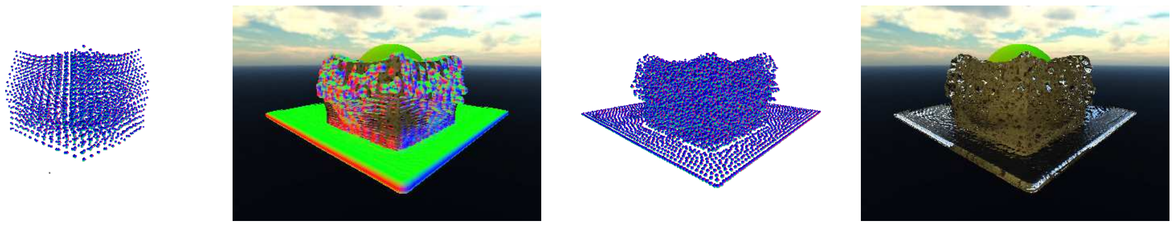

Figure 1: *(a) Sponge nodes simulated with PBD, which also serve as control nodes for SPH, (b) superimposed sponge and fluid surfaces, (c) fluid particles simulated with SPH, (d) realistic visualization.*

approach. However, porous bodies and interpenetration of fluid and elastic material are not considered.

3. Our proposal

We combine SPH fluid, SBD soft-body, and rigid-body mechanics to simulate interaction between sponges, solids, and fluids. All constituting subsystems were chosen for their proven real-time performance, and the possibility of extremely efficient parallel implementations. Figure 2 shows computation phases, data flow, and points of coupling between the simulation subsystems. In this section, we describe the underlying physical effects. We also summarize how the system can be visualized in real-time, ensuring the appearance of a porous surface for the sponge, and ray traced reflections and refractions for the fluid.

3.1. Controllable fluid simulation

Our fluid simulation is a classic SPH approach augmented with *control nodes*. These nodes introduce external forces. In general, these could be non-physical effects that drive the fluid to assume some desired shape, or follow the motion of an animated character, as in the work of Zhang et al. [ZYWL15]. In our case, we use control nodes to mediate forces from the porous medium.

The phase **Calculate mass density and pressure** is unchanged from classic SPH. Both quantities are computed at the particles using radial smoothing kernels, taking stiffness and saturation density parameters into account.

In the phase **Calculate forces**, we determine forces such as pressure, viscosity, gravity, and surface tension as per classic SPH. Additionally, a *suction* force is obtained following the template of the pressure force calculation, but using the control nodes instead of the fluid particles. The nodes are assigned a *control pressure* value. Thus, an artificial low-pressure volume can be created, which attracts fluid particles. The radius of the smoothing kernel may also be different from that applied to fluid particles. The effect of solid surfaces (like dynamic rigid bodies and enclosure walls) on the fluid can also be applied here, if they are formulated as elastic forces acting on fluid particles. This is consistent with the SPH approach not strongly enforcing fluid incompressibility. In phase **Apply forces and boundary constraints** fluid velocities and positions are updated using semi-implicit Euler integration.

In order to reduce a number of particle accesses for evaluating smoothed quantities, we use the single-detail-level scheme described by Winchenbach and Kolb [WK20]. The computationally expensive part of building this data structure in every frame are the two sortings, for which we use GPU radix sort. Both radix sort and the creation of the cell list and the hash lookup rely heavily on computing prefix sums. We broke the particle array into 32×32 pages, and utilized the shared GPU memory and wave intrinsics to efficiently compute these.

3.2. Sponge simulation with Parallelized Strain Based Dynamics

Our sponge simulation is based on SBD [MCKM14]. This means PBD constraints are formulated for tetrahedral elements. Position adjustments are iteratively applied to satisfy constraints on element shear and volume. Elasticity can be controlled using the stiffness parameter, which defines the strength of the constraint.

In phase **Calculate initial position change**, Euler integration is performed in order to find predicted positions. Then the **PBD Solver** applies both structural constraints (written for tetrahedra) and external constraints (contact with rigid bodies) iteratively. In theory, PBD constraints must be applied sequentially. However, real-time operation requires parallel execution. This is possible if the elements can be sorted into geometrically disjoint sets, with no two tetrahedra of the same set sharing a vertex. We accomplish such a setup using the *tetragonal disphenoid honeycomb*. The vertices are aligned on a *body-centered cubic* grid, which maps conveniently to two 3D arrays for GPU representation. Every vertex has 24 connecting elements, and 12 of those elements constitute a shape that tiles space when aligned on a cubic grid. This defines 12 classes of tetrahedral elements. If we further split them into even and odd elements in a 3D checkerboard pattern, we obtain 24 geometrically disjoint classes. Parallel GPU constraint resolution can be executed on all constraints in 24 passes without conflicts. Because of the identical shape of the original, undistorted elements, the *coefficient matrix* for deformation gradient computation is a global constant that applies in every class, when adjusted for orientation by swizzling and inverting axes appropriately. This reduces memory requirements and saves significant computation bandwidth.

In phase **Calculate velocity**, using the final particle positions, velocity is updated.

3.3. Interaction between different simulations

Sponge on fluid effects include capillary suction and drag. As explained in literature [FBGZ18], suction is derived from *pore pressure*. We assume the sponge structure is isotropic, and thus the

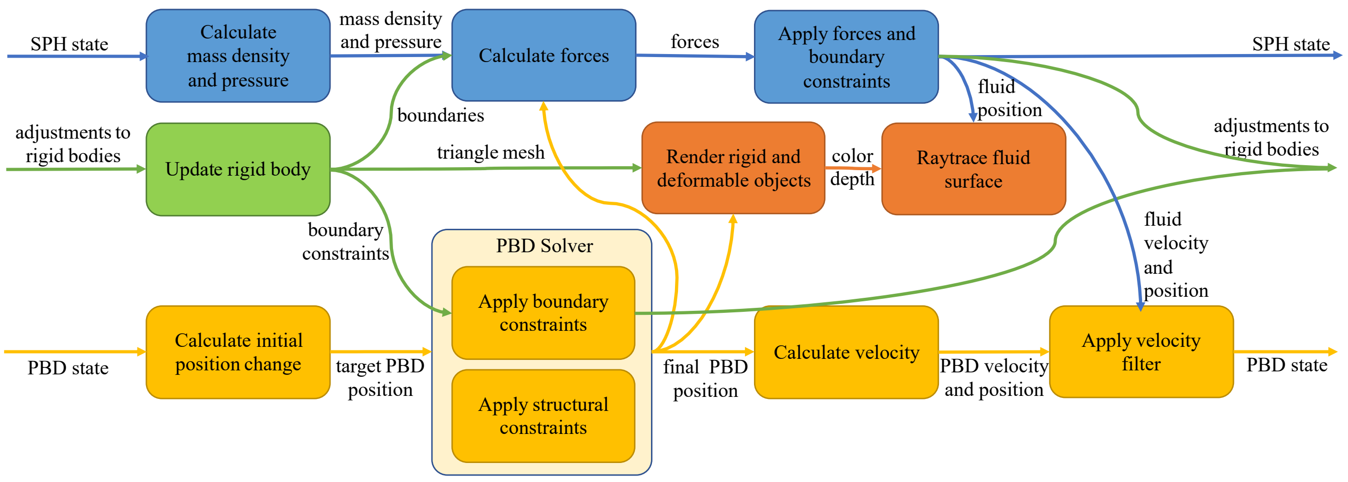
SPH state
Calculate mass density and pressure
mass density and pressure
Calculate forces
forces
Apply forces and boundary constraints
SPH state
fluid position
adjustments to rigid bodies
Update rigid body
boundaries
triangle mesh
Render rigid and deformable objects
color
depth
Raytrace fluid surface
adjustments to rigid bodies
boundary constraints
PBD Solver
Apply boundary constraints
Apply structural constraints
fluid velocity and position
PBD state
Calculate initial position change
target PBD position
final PBD position
Calculate velocity
PBD velocity and position
Apply velocity filter
PBD state

Figure 2: *Data flow in our combined system. Fluid simulation phases are blue, the rigid-body phases are green, the PBD simulation phases are yellow, and the rendering passes are orange.*

suction tensor reduces into a scalar. This is equivalent to having a constant negative pressure term modulated only by fluid saturation. This maps well to our controllable SPH method, if we consider PBD nodes as control nodes in the SPH simulation. In order to avoid the fluid surface defined by contained particles surrounding the sponge, the surface layer of nodes is ignored.

Drag is a velocity-dependent force that represents the medium dissipating kinetic energy from the fluid as defined by its permeability. If the sponge is moving, the relative velocity should be considered, making it possible for the moving sponge to induce motion in the fluid. A relative-velocity-based force term, viscosity, is already part of the SPH simulation. By computing a similar force over velocities of control nodes, replacing viscosity by the inverse of permeability, we apply a linearized drag model for the porous medium.

After the **PBD Solver**, the final positions and velocities are used in the **Calculate forces** phase as control particles. **Fluid on sponge** forces are opposing the sponge on fluid ones. Drag is a friction-like force, which is typically incorporated into PBD schemes using *velocity filters*, altering velocities before they are applied to positions. This computation is similar to the evaluation of the viscosity force in SPH, operating over fluid particles. This is performed in phase **Apply velocity filter**.

Given the suction force drawing the fluid into the sponge, and then the velocity filter transferring fluid speed to the porous solid, the sponge would start to move away from the fluid. Thus, maintaining plausibility, the suction force acting on the sponge cannot be neglected. In PBD, this has to be formulated in terms of constraints. A zero-distance constraint between PBD nodes and neighboring fluid particles is used, with stiffness empirically tuned to balance the suction factor.

Rigid bodies on sponge effects are contact constraints. **Rigid bodies on fluid** effects are formulated as repelling forces. Both **sponge and fluid on rigid bodies** are resolved using PBD in the **Update rigid body** phase, adjusting rigid-body pose to match constraints against fluid particles and solid nodes.

4. Rendering

Real-time rendering of fluids is challenging in itself, as reconstruction of the surface from a particle representation is expensive, and transparent fluids require tracing reflections and refractions. First, in the **Rendering rigid and deformable objects** phase, triangle mesh rasterization is used to display rigid bodies and the sponge. For this, a triangular mesh is generated using the outer PBD nodes. Note that, in general, the triangle mesh would not have to match the underlying disphenoid honeycomb, and it would be possible to use arbitrarily shaped meshes, disabling the nodes of the honeycomb outside of the shape during physical simulation. The resulting depth offset is written into the depth buffer for further composition.

A color and depth buffers laid down by rasterization are passed to the **Ray tracing of the fluid surface** phase. This implements recursive ray tracing. The surface is found by ray marching, evaluating density at marching points using the proximity search acceleration structure described in Section 3.1. At each marching point, its depth is compared against the solid depth. When the solid surface is hit, its color from the color buffer is retrieved. This is akin to *approximate ray tracing* using the rendered image of the solid objects as a *distance impostor* [SKALP05].

5. Results and discussion

We implemented our approach in C++, using a combination of D3D11, D3D12, and HLSL shaders. The simulation was tested on a PC with 32 GB of RAM and an Intel 4790k processor, using an NVIDIA GTX 1080ti graphics card. Based on our measurements, shown at Figure 3, the method works in real-time for lower particle and node counts, with acceptable visual quality. Higher particle counts are still interactive. Using 8192 fluid particles and 1024 sponge nodes the simulation and realistic rendering took 30 ms per

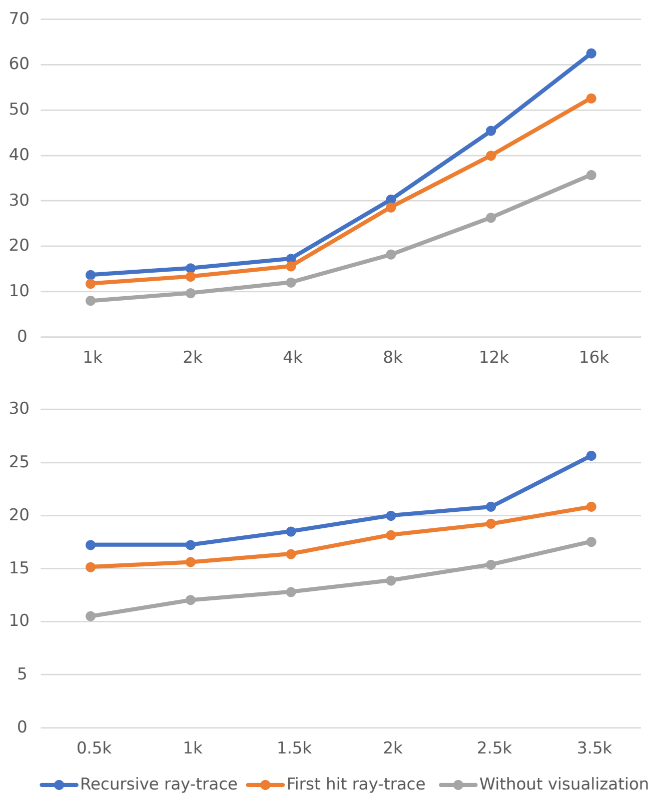

Figure 3: *Frame time (ms) dependence of (a) fluid particles with 1k control particles (b) control particles with 4k fluid particles, considering multiple visualization methods.*

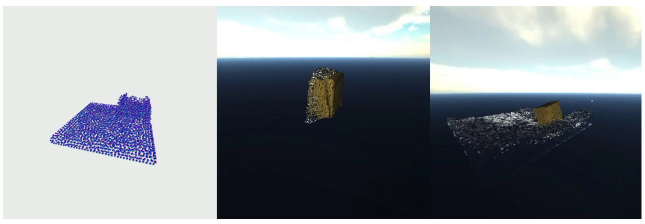

Figure 4: *(a) Sponge absorbing water, (b) sponge squished between planes, (c) fluid dragging sponge.*

frame. We investigated several test scenarios. **Sponge standing in water** tests the ability of the porous medium to pull in liquid due to pore pressure. How high the water rises depends on the strength of this force, i.e. the *porosity* of the material (Figure 4a). It is also important that the sponge stays stationary as the drag force and the suction constraint balance each other. **Ball falling on saturated sponge** (Figure 1) tests how fluid is pushed out from the sponge as its volume decreases. Material with lower *permeability* resist this effect more, slowing down the ejection of water. **Squishing the sponge between planes** is a similar scenario, but the planes move with constant velocity, and are not pushed back by the fluid or the sponge (Figure 4b). **Fluid dragging sponge** is a dam break scenario, where the drag force acting on the sponge is tested (Figure 4c). **Wringing the sponge** tests what happens when the volume is decreased once again, but also shows how centrifugal forces can spray fluid from a rotating body.

Our system plausibly handles capillary absorption of fluid into the sponge and discharge of it when compressed. The flow is able to carry the sponge, and flow in the porous medium is slowed down adequately. Wet sponge behaves differently than dry sponge without having to formulate this behaviour explicitly. We formulated our control nodes for SPH simulation, but using Position Based Fluids [MM13] would likely improve fluid incompressibility and performance. We have not demonstrated simulating multiple, arbitrarily shaped sponges, or larger scale flow. Handling collisions between soft bodies in real-time would be extremely challenging.

6. Acknowledgements

This work has been supported by OTKA K-124124. The research presented in this paper, carried out by BME, was supported by the Ministry of Innovation, and the National Research, Development and Innovation Office, within the framework of the Artificial Intelligence National Laboratory Programme.

References

[ANM*20] ABURUMMAN N., NAIR P., MÜLLER P., VANDERHAEGHE D., BARTHE L.: ISPH-PBD: Coupled Simulation of Incompressible Fluids and Deformable Bodies. *The Visual Computer* (05 2020). 1

[BMM17] BENDER J., MÜLLER M., MACKLIN M.: A Survey on Position Based Dynamics, 2017. In *Proceedings of the European Association for Computer Graphics: Tutorials* (Goslar, DEU, 2017), EG '17. 1

[FBGZ18] FEI Y., BATTY C., GRINSPUN E., ZHENG C.: A multi-scale model for simulating liquid-fabric interactions. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–16. 1, 2

[HZ21] HUANG S., ZHOU K.: Fluid-cloth coupling algorithm based on PBD and CCD. *Journal of Physics: Conference Series 1865*, 4 (apr 2021), 042066. 1

[LAD08] LENAERTS T., ADAMS B., DUTRÉ P.: Porous Flow in Particle-Based Fluid Simulations. *ACM Trans. Graph. 27*, 3 (aug 2008), 1–8. 1

[MCKM14] MÜLLER M., CHENTANEZ N., KIM T., MACKLIN M.: Strain Based Dynamics. In *The Eurographics / ACM SIGGRAPH Symposium on Computer Animation, SCA 2014, Copenhagen, Denmark, 2014* (2014), pp. 149–157. 1, 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 1, 4

[SI12] SZÉCSI L., ILLÉS D.: Real-Time Metaball Ray Casting with Fragment Lists. In *Eurographics* (2012). 1

[SKALP05] SZIRMAY-KALOS L., ASZÓDI B., LAZÁNYI I., PREMECZ M.: Approximate ray-tracing on the GPU with distance impostors. In *Computer graphics forum* (2005), vol. 24, Citeseer, pp. 695–704. 3

[WK20] WINCHENBACH R., KOLB A.: Multi Level Memory Structures for Simulating and Rendering Smoothed Particle Hydrodynamics. *Computer Graphics Forum 39* (08 2020). 1, 2

[XLF*19] XI R., LUO Z., FENG D. D., ZHANG Y., ZHANG X., HAN T.: Survey on Smoothed Particle Hydrodynamics and the Particle Systems. *IEEE Access 8* (2019), 3087–3105. 1

[ZYWL15] ZHANG S., YANG X., WU Z., LIU H.: Position-based fluid control. *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games, i3D 2015* (02 2015), 61–68. 1, 2