A Robust Volume Conserving Method for Character-Water Interaction Supplementary Material

Minjae Lee
Stanford University
mjlgg@stanford.edu

David Hyde
Stanford University
dab@stanford.edu

Kevin Li
Stanford University
kevli@cs.stanford.edu

Ronald Fedkiw
Stanford University
Industrial Light + Magic
rfedkiw@stanford.edu

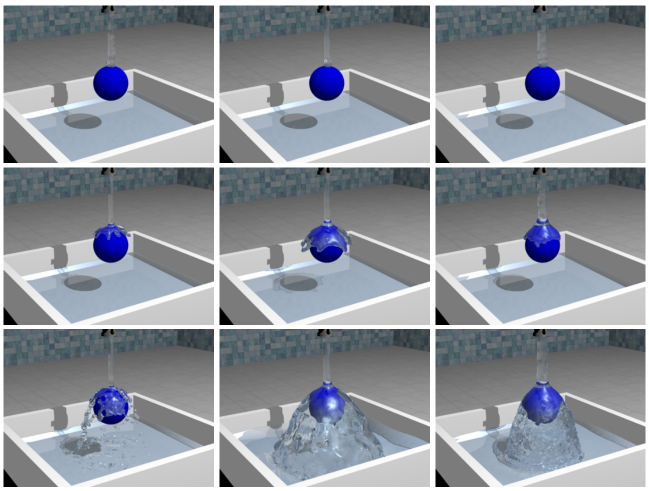

Figure 1: (Left Column) Our VOF method with a naive projection implementation which does not conserve volume. (Middle Column) Our VOF method with smear and pushout while replacing our velocity correction step with a standard Poisson solver. (Right Column) Our VOF method with proposed smear, pushout, and velocity correction steps. The middle and right columns conserve volume.

ACM Reference Format:
Minjae Lee, David Hyde, Kevin Li, and Ronald Fedkiw. 2019. A Robust Volume Conserving Method for Character-Water Interaction Supplementary Material. In *SCA '19:The ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '19), July 26–28, 2019, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3309486.3340244

In order to evaluate our method compared to other approaches and to explore possible extensions, we implemented a standard Poisson solver by assigning pressures on nodes similar to [Ando et al. 2013]. This implementation solves the inviscid, incompressible Navier-Stokes equations:

$$\partial \mathbf{u}/\partial t = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p/\rho + \mathbf{f}$$

while satisfying $\nabla \cdot \mathbf{u} = 0$ to enforce the divergence free condition for the velocity field without any advanced modifications (p is pressure, $\mathbf{f}$ is external forces). We ran two different flavors of this alternative; one is to completely replace our volume conservation scheme with the standard Poisson solver ignoring the volume conservation entirely within the projection, and the other is to replace only the velocity correction while keeping smear and pushout to conserve volume. Note that the smear and pushout steps transport fluid with its momentum, so oversaturated fluid velocity propagates to its neighbors. Thus, the second version spreads water outward more than the first version. We ran all implementations on the KDSM with the same setup, and the results are shown in the above Figure. In the Figure, we found that the right column is more desirable than the left because it conserves volume, and is faster and more robust than the middle column since we do not have to solve a linear system.

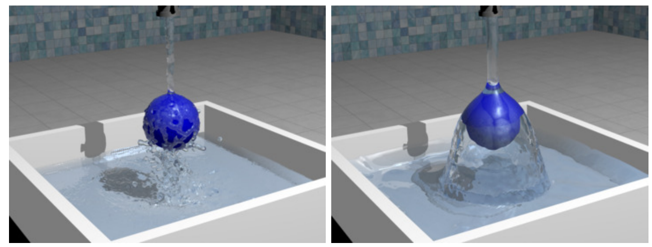

Figure 2: (Left) FLIP method on our ball example. (Right) Our method.

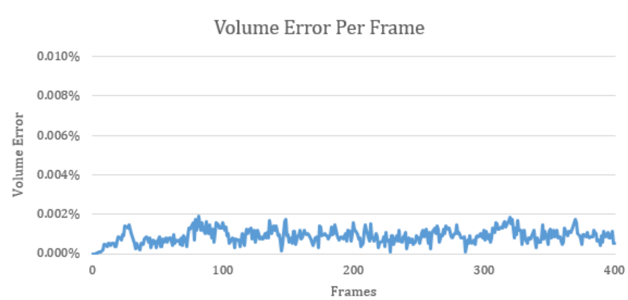

Figure 3: Volume error for example where a thin stream of water hits a ball.

SCA '19, July 26–28, 2019, LA, CA, USA

ACM ISBN 978-1-4503-6677-9/19/07…$15.00
https://doi.org/10.1145/3309486.3340244

Occasionally water stacking along boundaries can occur when VOF tetrahedra are in contact with solids.

This is due to our VOF volume conservation step distributing excess fluid and its momentum to neighboring tetrahedra, and this issue can be resolved either by increasing the resolution of the Eulerian grid to allow Eulerian fluid to contact the solid and using its full-fledged pressure solver as in the partitioned coupling section or by using a standard Poisson solver as discussed in the volume preservation section.

Algorithm 1 Pseudocode for Advection

```
// τ: tetrahedron, v: fluid velocity for τ, Δt
// Transports carry volume and associated momentum together
function ADVECTION
    BackwardAdvection() from the new mesh to the old mesh
    ForwardAdvection() from the old mesh to the new mesh
function BACKWARDADVECTION
    for each τ in the KDSM do
        backtraced τ = Backtrace(τ, −v, Δt)
        point samples = GeneratePointSamples(backtraced τ)
        for each point sample p in point samples do
            if p lies within a tetrahedron τ_old with water then
                Preprocess for conservative advection
            else if p falls under the Eulerian water then
                Transport water from the Eulerian grid
    for each τ in the KDSM do
        for each point sample p in point samples do
            Transport water with preprocessed conservation terms
function FORWARDADVECTION
    for each τ in the KDSM do
        traced τ = Backtrace(τ, v, Δt)
        point samples = GeneratePointSamples(traced τ)
        for each point sample p in point samples do
            if p lies within the KDSM then
                Transport water to an appropriate tetrahedron
            else
                Transport water to an appropriate Eulerian grid
function BACKTRACE(τ, v, Δt)
    Trace nodes of t backward in time with v and Δt
    for each Traced node with position x do
        if Collide(x, any solid surface) then
            Clamp x with collided location
function COLLIDE(x, y)
    return True if x collides with y, False otherwise
function GENERATEPOINTSAMPLES(τ)
    point samples = QuadratureFormula(backtraced τ)
    volume = volume of τ / number of point samples
    attach volume to each point samples
    return samples with volumes attached
```

Algorithm 2 Pseudocode for Volume Preservation

```
// τ: tetrahedron
// Transports carry volume and associated momentum together
function VOLUME PRESERVATION
    Smear()
    Pushout()
    VelocityCorrection()
function SMEAR
    for each τ in the KDSM do
        if τ is not on the boundary and is oversaturated then
            Distribute excess fluid equally to τ's neighbors
function PUSHOUT
    for each τ in the KDSM in the order of lowest rank to highest do
        if τ is oversaturated then
            if τ is on the boundary then
                Push water out of the KDSM as particles
            else
                Distribute excess water as much as possible to its face neighbors equally as long as they are not oversaturated
                Distribute the remaining excess water as much as possible similarly to face neighbors with strictly higher rank
                Distribute the remaining excess water to tetrahedra which are precomputed
function VELOCITYCORRECTION
    Allocate a Boolean per tetrahedron and initialize to false
    for each τ in the KDSM do
        if τ is a cut cell and has water then
            set τ's Boolean to true
    for each τ in the same order as in the pushout do
        if τ has a face neighbor with lower rank and which is fully saturated and has Boolean set to be True then
            Clamp the normal velocity and set Boolean to be True
```

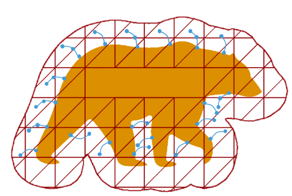

Figure 4: Yellow bear mesh is enclosed by the red KDSM, which embeds hairs via blue particles.

REFERENCES

R. Ando, N. Thürey, and C. Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph. (Proc. SIGGRAPH 2013)* (July 2013).

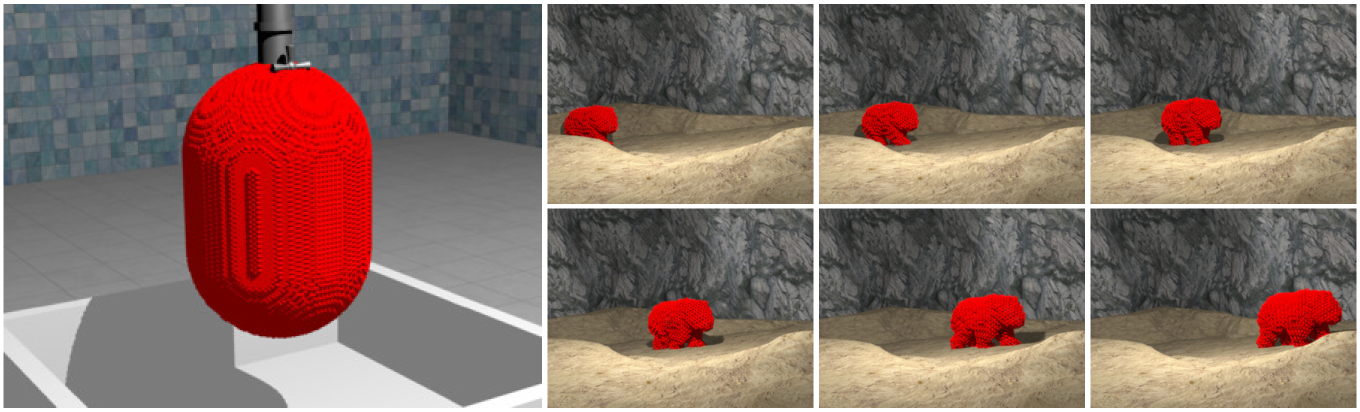

Figure 5: (Left) A KDSM mesh around the ball. (Right) A sample animation showing the KDSM skinned to follow an animation of a bear walking on a shore.