A SKINNED TETRAHEDRAL MESH FOR HAIR-WATER INTERACTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Minjae Lee
December 2018

Abstract

This dissertation presents kinematically deforming skinned mesh (KDSM), a novel data structure that serves as a foundation to implement hair-water simulation system for hair animation, robust volume-conserving water simulation, and hair-water interaction.

Both hair and water simulation techniques have become very popular in the entertainment industry, especially in motion pictures; however, these techniques suffer from limited scalability, difficult artist control, and a lack of a flexible framework that incorporates multiple methods. Researchers have adopted approximative approaches for hair simulation, such as volumetric hair and guide hair, which are more efficient and easier to control than simulating each individual hair, but the benefits of these are situational and limited, and the results are often of lower quality. For water simulation, spatially adaptive methods such as Adaptive Mesh Refinement (AMR), octree data structures, lattice based tetrahedral methods, and Chimera grids are used for efficiency, but these lack robustness, are difficult to implement/control, and become less efficient in many scenarios. In order to address these problems, we developed KDSM, a skinned tetrahedral mesh, which contains air volume around the character, includes inertial effects and deformations of the character animation, and maintains a consistent topology. Based on such features of the KDSM, we present a hair-water framework that efficiently computes hair, water, and hair-water phenomena while offering a powerful control for artists and providing a flexible framework to integrate multiple methods with ease.

After the introductory chapter, we discuss our novel hair animation technique and hair-water interaction scheme. We provide the following layering framework which is

well suited for an iterative feedback loop of a creative process, offering a straightforward and powerful artist control by separating bulk hair motion and intricate hair motion. For an input character animation, we initialize a tetrahedral mesh, KDSM, in a sculpt/normalized/T-pose and generate a corresponding animation sequence of KDSMs via morph that handles any given constitutive model, e.g., mass spring. With the animated KDSM sequence, we apply kinematic skinning by embedding hair particles to follow the KDSMs to achieve bulk hair motion, and additionally apply dynamic skinning by applying more deformation on the copied KDSMs when external forces, such as wind and water drag, are needed. Then, if intricate hair motion is required, we use either blendshape hair for precomputed intricate hair motion within the KDSM or individual hair simulation for dynamic motion can be run with any individual hair model (adaptively if desired). This chapter ends with a discussion of how we run our novel hair-water solver with a hair porosity scheme to handle hair-water interactions in the Eulerian grid.

In the next chapter, we present a robust volume-conserving character-water interaction. A coarse Cartesian Eulerian grid is used to capture bulk water motion, and our novel volume-conserving volume-of-fluid method in arbitrary Lagrangian-Eulerian mesh derived from the KDSM is used to achieve intricate water motion with high adaptivity and volume preservation around the character. We precompute various auxiliary information such as adaptivity via subdivision, topological information, and porosity to improve the simulation time as well as the robustness. Also discussed is a fast, robust, and simple partitioned approach to two-way couple our two distinct fluid solvers specializing in their own domain of interest. This approach allows us to highly specialize our volume-of-fluid solver so that we can control fluid motion more easily, such as adhesion effects and anisotropic porosity for hair whereas the background fluid solver focuses on handling bulk water surfaces using level sets. Furthermore, this chapter presents our novel water surface reconstruction, taking advantage of advection step of the particle level set method to merge output of our two distinct water solvers in a time-coherent manner. Finally, we demonstrate hair-water interaction in a setting similar to that discussed in the previous chapter with improved visual quality through our novel water framework.

Acknowledgments

First, I would like to thank my advisor, Ron Fedkiw for his continuous support. Your vision of practical applications of research had a profound impact on my life, not to mention your advice and criticisms. I am deeply thankful for your support for giving me the freedom to explore various ideas and pushing me to develop them fully.

I also would like to thank my readers, Kayvon Fatahalian and Philip Levis for their insightful feedback. I was an undergrad research assistant for Kayvon at CMU, and when I was picking a Ph.D. program, Kayvon's advice was a tie breaker for me to choose Stanford over the other school. I would not have met my wife if I had chosen otherwise. I am immensely grateful for this and also for your support starting from my senior year at CMU until the end of my Ph.D.! I am also grateful to have Gordon Wetzstein and Dorsa Sadigh for serving in my Ph.D. defense and providing interesting perspectives.

My best man, Sae Young Lee, thank you for the many, many sleepless gaming nights. I cannot believe that we became adults!

I had enormous fun with my friends at Stanford. I would like to thank Ed Quigley for helping me with my writing, Winnie Lin for voiceacting for my paper, and Michael Bao for recommending me CK2 and EU4. Matthew Cong, thank you for your mentorship during my Ph.D. Aric Bartle, thank you for helping me with my defense. Your startup is awesome. I enjoyed working with Bo Zhu, Saket Patkar, Rahul Sheth, Kevin Li, and Mridul Aanjaneya. I would like to thank Minhyuk Sung for lending me a couch and ordering food when I was immobalized after getting hit by a car. I would also like to thank Sean Liu, Jenny Jin, Yue Yu, Linhai Qiu, and Wenlong Lu.

Jongwoo Park, thank you for your friendship and support. You should do a

Ph.D. too! Minyoung Lee, thank you for the Steam money. Good luck with your Ph.D. Jung-In Kim, thank you for your support and many dinners. I would like to thank EteRNA members next. Jeehyung Lee, thank you for offering my first research position in CMU. I hope everything is well with your startup. Adrien Treuille, thank you for your advice and support during my undergrad years. I would like to thank Rhiju Das for his support since my undergrad. We should do another EteRNA retreat. I should not forget my academic advisor at CMU. Mark Stehlik, thank you for your helpful advice and sharp criticisms when I enrolled in way too many courses and tried to do research at the same time.

I would like to thank Samsung Scholarship and Max Planck for their financial support for my research.

I will always remember my family for my success at Stanford. I would like to thank my mom and dad for their love and unconditional support, both mentally and financially. I am here today because my parents respected me as an indepedent entity and let me make my own decisions with my life. My sister also helped me with people skills. You are the best sister ever. I would like to thank my grandparents for their love and support. May my grandpa rest in peace. Still love you and miss you. My in-laws, Seok Jai Lee, Yeong Suk Yang, and Jongchan Lee, thank you for your encouragement and kind words. Same goes for the rest of my family members and cousins.

Finally, I would like to thank the most important person in my life - my wife, Yujin Lee, for her love and support. I cannot describe how much I love you and feel lucky to be your husband. Good luck with your Ph.D.! You are the cutest and smartest!

I dedicate this dissertation to Yujin, who inspires me every day to be a better person.

Contents

Abstract **v**

Acknowledgments **vii**

1 Introduction **1**

2 Hair Animation and Hair-Water Interaction **5**

2.1 Introduction 6

2.2 Previous Work 9

2.3 Layering Framework 10

2.3.1 Skinning Kinematic Hair 11

2.3.2 Simulating Dynamic Hair 14

2.3.3 Blendshape Hair 21

2.3.4 KDSM for Hair-Water Interaction 23

2.4 Examples 25

2.5 Conclusion 29

3 Character-Water Interaction **30**

3.1 Introduction 30

3.2 KDSM 35

3.3 KDSM Fluids 37

3.3.1 Precomputation 37

3.3.2 Advection 38

3.3.3 Volume Conservation 40

3.3.4 Adhesion 46
3.4 Partitioned Coupling 48
3.4.1 Particle Generation 53
3.4.2 Rendering 53
3.5 Hair-Water Interaction 56
3.6 Results and Discussion 56
3.7 Conclusions 59

4 Conclusions **61**
4.1 Rendering 62
4.2 Hair Framework Improvement 63
4.3 Cloth Simulation 63
4.4 Smoke Simulation 63
4.5 Tree Simulation 64
4.6 Artist Control for Water Simulation 64
4.7 Real-Time Hair/Water for Games 64
4.8 Adaptive Mesh for Neural Network 65

Bibliography **66**

List of Tables

2.1 List of examples with their resolutions and average runtime per frame. Note that the first layer, generating the KDSM, is not included. . . . 27

List of Figures

2.1 Given a skinned animation (top left), we skin a tetrahedral mesh to follow that animation while including the effects of inertia, deformation, pinching, etc. (top right). Subsequently, hair embedded in the tetrahedral mesh moves and deforms to follow that mesh with the potential for additional simulation (bottom left). The kinematically deforming tetrahedral mesh facilitates the use of a blendshape system allowing one to specify hairstyles procedurally as hair is exposed to water (bottom right). Final result, 540k hairs rendered (far right). 7

2.2 Our framework prescribes a workflow that is flexible with respect to its underyling constitutive model. In this diagram, orange components are computationally intensive stages of our pipeline and yellow components require iterations of artistic control. Starred components are optional depending on the effects desired for the animation. 8

2.3 (Top) We create a kinematically deforming skinned mesh (KDSM) that follows the animation and includes the effects of inertia, deformation, swinging motions, etc. (Bottom) Subsequently, any reference frame hairstyle can be skinned throughout the animation by simply following its embedding in the KDSM via barycentric coordinates. 9

2.4 Treating hair volumetrically, a simulated dynamic tetrahedral mesh is attached to the KDSM for structure and support, and sways back and forth underwater using the water velocity to apply drag. (Top) No blendshape hair. (Middle) Blendshape hair. (Bottom) Blendshape hair with water. 10

2.5 (Top left) KDSM. (Top right) No simulation. (Bottom left) Simulation with weak zero-length spring attachments to the KDSM. (Bottom right) Simulation with stronger zero-length spring attachments to the KDSM. 13

2.6 Starting from the rest hairstyle (top left), individual hair and volumetric hair simulations as well as procedural techniques were used to create clumped (top right), sagged (bottom left), and matted (bottom right) hairstyles, which were then all bound to the animation sequence from Figure 2.3 using that KDSM. 14

2.7 Our framework supports selective activation of individual hairs for simulation in order to only activate hairs that are undergoing collision. Here, fully kinematic hairs are shown in yellow while hairs activated for individual simulation are shown in green. 15

2.8 (Left) Hairs are individually simulated for high fidelity collision handling. (Right) Individual hair simulation is also used for long hairstyles. Both examples benefit from structure and support provided by the KDSM. 17

2.9 Visualization of structure contributing to the final hair shape: the simulated hair (black curve) is connected to embedded positions within the KDSM (red box) and to shape-preserving tetrahedra (blue boxes) by zero-length springs (yellow and green, respectively). 18

2.10 (Top row) Using the hairstyles from Figure 2.6 and the KDSM from Figure 2.3 along with our blendshape system, we make a creature grow progressively wetter as it walks. (Bottom row) Close ups of top row. 19

2.11 (Top) First a water simulation is performed, ignoring the hair. (Bottom) Then in a post-process, a procedurally generated wet map is used in the blendshape system to locally (in space and time) interpolate between various hairstyles such as those in Figure 2.6. 20

2.12 (Top) Creature walking out of water. (Bottom) Close ups of top row. 22

2.13 Water poured over creature. 24

2.14 (Top) Creature geometry and KDSM. (Left) No KDSM. (Right) Same frames as on the left, except using KDSM for increased drag on the water from the creature. 26

3.1 (Left) Whale breaching with the PLS method using automatically generated removed particles for spray. Very little of the water volume follows the whale's motion because of volume loss on the relatively coarse Eulerian background grid. (Right) Using the same Eulerian grid, our ALE based VOF method on the KDSM produces much more visually interesting sheeting and spray effects. 31

3.2 (Top Left) A KDSM mesh around a whale in a normalized pose (also known as T-pose or rest pose). (Top Right) A sample animation showing the KDSM skinned to follow an animation of a whale breaching. (Bottom Left) A KDSM mesh around the ball. (Bottom Right) A sample animation showing the KDSM skinned to follow an animation of a bear walking on a shore. 32

3.3 Same as Figure 3.4, but using an even smaller water stream accentuating the benefits of our approach especially when considering volume conservation. 33

3.4 (Top Left) A standard PLS simulation on a relatively coarse Eulerian grid. (Top Right) Our ALE based VOF method on the KDSM achieves better water sheeting and volume conservation using the same Eulerian grid. (Bottom Row) Applying adhesion forces to both simulations produces the desired clinging to the ball with our method but has almost no effect on the PLS simulation. 35

3.5 A backtraced triangle shown in red uses each of its 10 quadrature point samples to transport water from the old mesh to the new one. Here, we render both the old and the new mesh in the same location assuming that the KDSM is not moving for simplicity of depiction. Each red dot point sample would attempt to remove 1/10 of the area of the red triangle from the specific yellow triangle that it is interior to. 39

3.6 The triangles cut by the green solid surface region would be assigned rank 0, their node neighbors shown in red would be rank 1, and their node neighbors shown in yellow would be rank 2. The arrow shows how a rank 1 triangle needs to look non-locally in order to find a rank 2 triangle that it can deposit excess water into. 41

3.7 Our ALE based VOF method provides robust adhesion control to produce various effects. All four of these examples are obtained by merely varying adhesion effects. 42

3.8 A whale breaches out of the water. (Top Row) Visualization of the Eulerian water. (Second Row) VOF tetrahedra water is rendered in pink. (Third Row) Particles are shown in yellow. (Fourth Row) Final rendering. 44

3.9 (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. 47

3.10 Top row compares the particle positions obtained with uniform versus jittered sampling emphasizing how well our eyes capture structured information (even when we do not want them to). Bottom row compares the two approaches for an actual simulation. 50

3.11 Our anisotropic porosity model is implemented to influence the VOF method on the KDSM accounting for both limited volume fraction and drag/adhesion yielding visually compelling results. 51

3.12 (Left) Whale breaching with the PLS method on high resolution grid. The whale pulls very little water along with it. (Middle) FLIP method, which also produces similar amount of sprays. (Right) Our method pulls more water into the air with the whale, producing interesting effects such as sheets and sprays. 52

3.13 A bear walks out of a pool onto land, still carrying and dripping a large amount of water from its fur. Our anisotropic porosity model accounts for the correct volume fraction of water in the fur and uses adhesion to pull that water out of the pool with the bear, subsequently slowing dripping the water out of the fur. This example emphasizes the efficacy of the adaptivity of the KDSM as well as the ability to preserve volume and avoid disappearing water with our VOF method. 54
3.14 A close-up of the bear example, showing water sticking to fur and splashes generated from our VOF method. 55
3.15 (Left) Anisotropic smoothing kernel in our ball example. (Right) Our surface reconstruction method. 56
3.16 Volume error for example where a thin stream of water hits a ball (see top right of Figure 3.3). 58
3.17 (Left) FLIP method on our ball example. (Right) Our method. . . . 59

Chapter 1

Introduction

In the entertainment industry, hair and water effects are predominant features needed to express an artistic vision. While both technologies have developed tremendously over the last decade, their uses are still limited to high-budget projects, such as motion pictures and AAA games, and they require skilled specialists to spend a vast amount of time and resources to achieve the desired effect.

Thus, hair and water effects are hot topics in computer graphics, and much progress has been made to improve the aforementioned issues. Many researchers have approached hair simulation in a volumetric fashion to approximate hairs [46, 10, 75], and more specialized guide-hair variants [76, 24, 96] to improve efficiency along with a better artist control. However, because the volumetric approach is approximative, the results are often physically implausible, offer only limited control, and suffer from numerous approximation artifacts. Alternatively, [81] simulated each individual hair strand to obtain impressive results, but at the cost of time and computation power, which makes this approach feasible only in a heavyweight cluster environment without much artist control.

Water simulation has many variants with similar objectives, and many researchers focused on achieving spatial adaptivity to improve the simulation time as in Adaptive Mesh Refinement (AMR) [89], octree data structures [61], [60], [1], lattice based tetrahedral methods [25], [11], [8], and Chimera grids [33], [34], but there are many known problems, such as instability and frequency of remeshing, high communication costs, and domain decomposition issues because of a large number of small patches. Additionally, researchers have implemented various hybrid methods such as the particle level set (PLS) method [35], [36], including spray particles [62] in order to integrate particle and grid representations, maximizing the benefits that can be gained from each method based on the technical insight that we can use different models for different scales. Even though both hair and water simulations improved significantly since their inception from the previous work, many challenges remain that prevent hair and water simulation technologies from being adopted more widely in the industry.

The work presented in this dissertation addresses these challenges with a novel data structure, kinematically deforming skinned mesh (KDSM). The KDSM can deform along with the solid surface while maintaining a consistent topology for the surrounding air volume instead of the grid which is fixed in space or particles which do not have any connectivity. Specifically, we have developed a novel hair-water interaction framework that improves scalability, offers a powerful artist control, and provides a flexible framework capable of integrating multiple methods as follows:

Scalability: The KDSM, initialized as a coarse mesh, improves the scalability of hair and water simulation by (1) supporting precomputation of numerous auxiliary information (e.g. blendshape hair weights, adaptivity, topological information, adhesion coefficients, and porosity), (2) allowing the framework to be layered with bulk motion and intricate motion in a separate manner to use specialized solvers for each scale, and (3) providing a kinematic support for the simulated meshes that are anchored to use weak springs.

Artist Control: The KDSM improves artist control by (1) providing a visual guide for artists, (2) applying the desired amount of KDSM deformation to the simulated meshes, (3) containing air volume for hair particles to move freely instead of being constrained to lower-dimensional elements, and (4) achieving both adaptivity and detailed boundary conditions for the water simulation.

Flexibility: The hair-water framework supports the layering and the partitioned coupling based on the KDSM because of its agnostic nature towards simulation, which does not dictate any particular approach to simulation, making our framework very flexible.

The framework consists of two novel components: a hair animation-focused component and a robust volume-conserving water component. In addition, a hair-water solver is implemented for both cases.

In the following chapter, we focus on hair animation and hair-water interaction using KDSM. The separation of bulk hair motion and intricate hair motion allows us to develop a novel layering framework for a straightforward and powerful feedback loop for artists to iterate with the hair pipeline instead of having to rerun the entire simulation from scratch per iteration. For a given character animation input, we create a KDSM per frame via morph, which is flexible enough to handle any constitutive model for tetrahedral mesh (we used mass spring and zero length spring model, but finite element method or position based dynamics is also viable option). If the bulk hair motion is desired, we use kinematic skinning by embedding hair particles into the KDSM, and dynamic skinning is added by duplicating the KDSM with the corresponding constitutive model and allowing it to deform with external forces, such as wind and water drag, while being attached to the original KDSM with zero length springs. For the intricate hair motion, we use blendshape hair to animate detailed hair motion precomputed within the KDSM by allowing hair particles' hardbound locations to drift in order to handle clumped, sagged and matted hairstyles, whereas simulation of each individual hair (one can choose any individual hair simulation method) with zero length spring anchoring such hairs to the KDSM is used when dynamic hair simulation of individual hair strands is required, e.g., in collision. We demonstrate our hair-water solver using porosity for hair and the PLS method.

In the next chapter, we illustrate a robust volume-conserving water simulation framework for character-water interaction. We implemented a novel volume-conserving volume-of-fluid (VOF) method in an arbitrary Lagrangian-Eulerian (ALE) mesh derived from the KDSM, coupled with a coarse background Cartesian Eulerian grid employing the PLS method. We prebake our ALE mesh prior to actual simulation and precompute numerous auxiliary data, such as adaptivity through subdivision, topological information, and porosity, based on the fact that the KDSM is computed from kinematically deforming character. We achieve high spatial adaptivity around the character from our densely subdivided ALE mesh from the KDSM capturing intricate water motion, while the background grid is coarse by design for bulk water behavior. Our approach is fast, robust, and streamlined due to our simple partitioned approach to two-way couple our two distinct fluid solvers. Our VOF method preserves volume, which allows us to robustly implement adhesion and anisotropic porosity for hair to control water effects as well as to implement hair-water interaction. Last, we post-process the simulated water surface with our novel water surface reconstruction scheme, using an advection-only modification of PLS simulation in a refined grid with merged data from both Eulerian grid and ALE mesh to retain temporal coherency with the fewer bumpy artifacts that are present in smoothing kernel methods.

Chapter 2

Hair Animation and Hair-Water Interaction

In this chapter, we propose a novel framework for hair animation as well as hair-water interaction that supports millions of hairs. First, we develop a hair animation framework that embeds hair into a tetrahedralized volume mesh that we kinematically skin to deform and follow the exterior of an animated character. Allowing the hairs to follow their precomputed embedded locations in the kinematically deforming skinned mesh already provides visually plausible behavior. Creating a copy of the tetrahedral mesh, endowing it with springs, and attaching it to the kinematically skinned mesh creates more dynamic behavior. Notably, the springs can be quite weak and thus efficient to simulate because they are structurally supported by the kinematic mesh. If independent simulation of individual hairs or guide hairs is desired, they too benefit from being anchored to the kinematic mesh dramatically increasing efficiency as weak springs can be used while still supporting interesting and dramatic hairstyles. Furthermore, we explain how to embed these dynamic simulations into the kinematically deforming skinned mesh so that they can be used as part of a blendshape system where an artist can make many subsequent iterations without requiring any additional simulation. Although there are many applications for our newly proposed approach to hair animation, we mostly focus on the particularly challenging problem of hair-water interaction. While doing this, we discuss how porosities are stored in the kinematic mesh, how the kinematically deforming mesh can be used to apply drag and adhesion forces to the water, etc.

2.1 Introduction

The seminal rendering of the teddy bear in [52] immediately piqued a great deal of interest in hair animation and simulation, see [78, 9]. Since then hair simulation has progressed a great deal and has been used as a signature effect in order to create many iconic characters such as Sulley in *Monsters, Inc.*, the Incredibles [75], Rapunzel in *Tangled* [97, 84], Merida in *Brave* [49], Elsa in *Frozen* [100, 85], Puss in Boots, many of the endearing creatures in *Zootopia*, and most recently Moana.

We provide a novel framework which supports millions of hairs as well as hair-water interaction with layering and flexibility so that one can easily control the hair in a straightforward manner as well as implement it based on their method of choice rather than requiring our approach to be used for all pieces of the framework. Our key observation is that once a character animation is finalized, one can utilize a volumetric approach to hair in order to skin the volumetric region of air that will subsequently contain the hair, approximating the bulk hair motion. This skinned air provides dramatically better three-dimensional volumetric structure and support for subsequent hair simulation than that provided by a lower-dimensional (i.e. 2D) surface skin of the character. For a still character the mesh would be static and rigid, but for an animated character the kinematically deforming mesh would be skinned to include both inertial effects and deformations resulting from those inertial effects (even swinging motions). Moreover, the kinematically deforming mesh can be made to include effects due to deformations in the shape of the character, volume preservation, collisions and self-collisions, folding and pinching, etc. Since the bulk effects of the air region around the character are mostly induced by the pre-prescribed character animation, it makes little sense to simulate this region over and over while trying to obtain the desired subtle, creative, and/or stylistic behavior of the hair within this volume. Instead, hair which is bound to follow our kinematically deforming mesh automatically includes all the aforementioned volumetric effects without the need

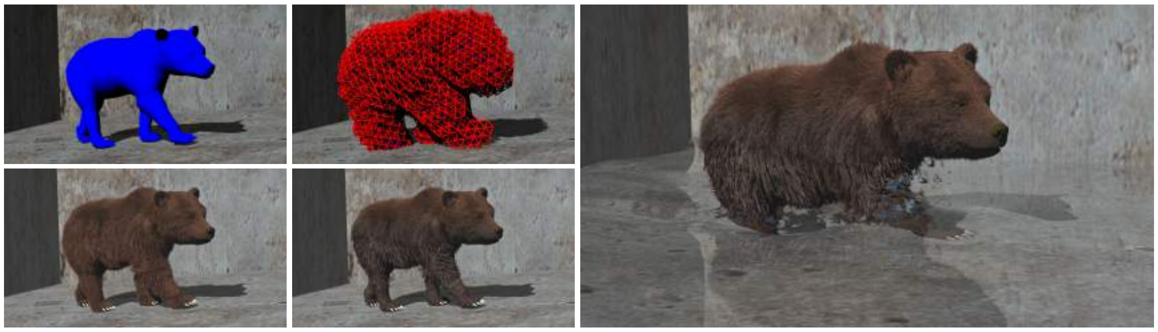


Figure 2.1: Given a skinned animation (top left), we skin a tetrahedral mesh to follow that animation while including the effects of inertia, deformation, pinching, etc. (top right). Subsequently, hair embedded in the tetrahedral mesh moves and deforms to follow that mesh with the potential for additional simulation (bottom left). The kinematically deforming tetrahedral mesh facilitates the use of a blendshape system allowing one to specify hairstyles procedurally as hair is exposed to water (bottom right). Final result, 540k hairs rendered (far right).

for any simulation at all, and any subsequent simulation would only be required for computing deviation from this kinematically skinned guide mesh.

There are many circumstances where hair bound to the kinematic mesh is already visually pleasing since the aforementioned volumetric effects are already included. Moreover, when subsequent simulation is desired, the kinematic mesh provides for dramatic gains in efficiency. For example, one can still simulate a dynamic tetrahedral mesh, but this dynamically simulated mesh would then be attached to the kinematically deforming skinned mesh with zero-length springs providing structure and support so that only very weak springs would be required internal to the dynamic tetrahedral mesh. Similarly, individual hairs can be anchored barycentrically to the kinematically deforming skinned mesh so that they inherently retain much of their shape and structure while using only a relatively weak, and thus efficient to simulate, mass-spring system internally. In addition to the benefits our kinematically deforming skinned mesh provides for the kinematic animation and dynamic simulation of hair, it also provides a convenient, time-coherent parameterization of the free space around the character which readily lends itself to the implementation of an extremely efficient blendshape hair system enabling high-level artistic and directive

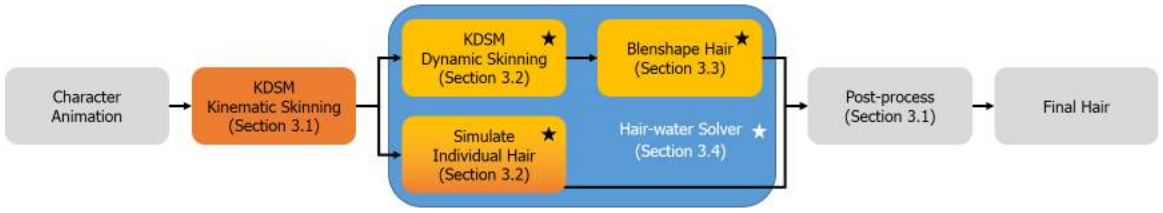

Character Animation
KDSM Kinematic Skinning (Section 3.1)
KDSM Dynamic Skinning (Section 3.2) ★
Blenshape Hair (Section 3.3) ★
Simulate Individual Hair (Section 3.2) ★
Hair-water Solver (Section 3.4) ★
Post-process (Section 3.1)
Final Hair

Figure 2.2: Our framework prescribes a workflow that is flexible with respect to its underyling constitutive model. In this diagram, orange components are computationally intensive stages of our pipeline and yellow components require iterations of artistic control. Starred components are optional depending on the effects desired for the animation.

control by the animator. This is especially useful in handling intricate hair motion such as wetting and clumping of hairs. Finally, noting that [79] demonstrated the benefits of using an Eulerian grid for hair-water interaction, we demonstrate some of the benefits that our kinematically deforming skinned mesh (KDSM) also provides for hair-water interaction, albeit in an arbitrary Lagrangian-Eulerian (ALE) rather than Eulerian framework.

The first contribution of this paper is a layering of the framework, which provides a straightforward and powerful artist control. The layering provides a separation of the treatments of bulk hair motion and intricate hair motion, and it allows an artist to have a highly efficient iterative feedback loop while maintaining stunning visual quality. The KDSM and blendshape hair systems enable this because we allow hard-bound locations to drift instead of being constant throughout the animation. Our second contribution is a unified and flexible framework that can handle both volumetric and individual simulation of hairs, and use readily available physical models along with the existing character animation to simulate hairs. The third contribution of our method is an interface between kinematic hair and simulated hair via a novel shape-preserving structure that supports the original shape of the hair. The fourth contribution of the KDSM is that it supports hair-water interactions and facilitates effects such as adhesion and drag on water. An overview of our method is given in Figure 2.2.

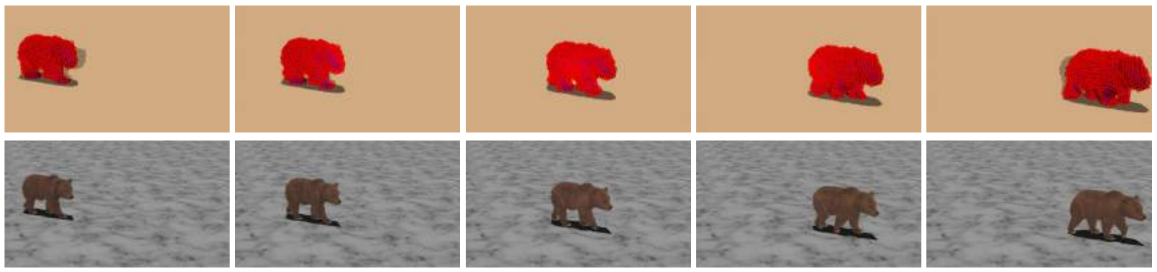


Figure 2.3: (Top) We create a kinematically deforming skinned mesh (KDSM) that follows the animation and includes the effects of inertia, deformation, swinging motions, etc. (Bottom) Subsequently, any reference frame hairstyle can be skinned throughout the animation by simply following its embedding in the KDSM via barycentric coordinates.

2.2 Previous Work

Hair simulation is a hot topic in computer graphics, and various authors have approached hair simulation by explicitly assuming that hair acts as a volume, see e.g. [46, 10, 75]. Methods that use guide hairs, see e.g. [76, 24, 96, 27], also assume that they are simulating hair as a volume, using this assumption to create interaction rules for simulated guide hairs and to interpolate renderable hairs from guide hairs. That is, guide hairs are just the degrees of freedom chosen to represent the hair volume, something exploited by [23] to obtain impressive results. [22] builds on this approach using an adaptive simulation scheme to handle hair-solid collisions. Alternatively, some authors aim to simulate every hair, see e.g. [81, 63, 31, 49, 12], in which case it becomes important to consider physical interactions between actual hairs (see e.g. [53]) as opposed to interactions based on volumetric continuum assumptions for guide hairs.

Motivated by [83, 71, 101], we use a volumetric tetrahedral mesh in the air region in order to treat both collisions and hair-hair interactions. However, our approach is more similar to [91, 92], where the hair was embedded in a simulated volumetric lattice enclosing the head. [15] pointed out that hair simulated in this fashion too closely follows the continuum and subsequently proposed a two-layered approach similar to [54] and [70]. We instead address this by utilizing a hybrid solids simulation framework [82] with soft bindings that allow our hairs to drift away from their embedded

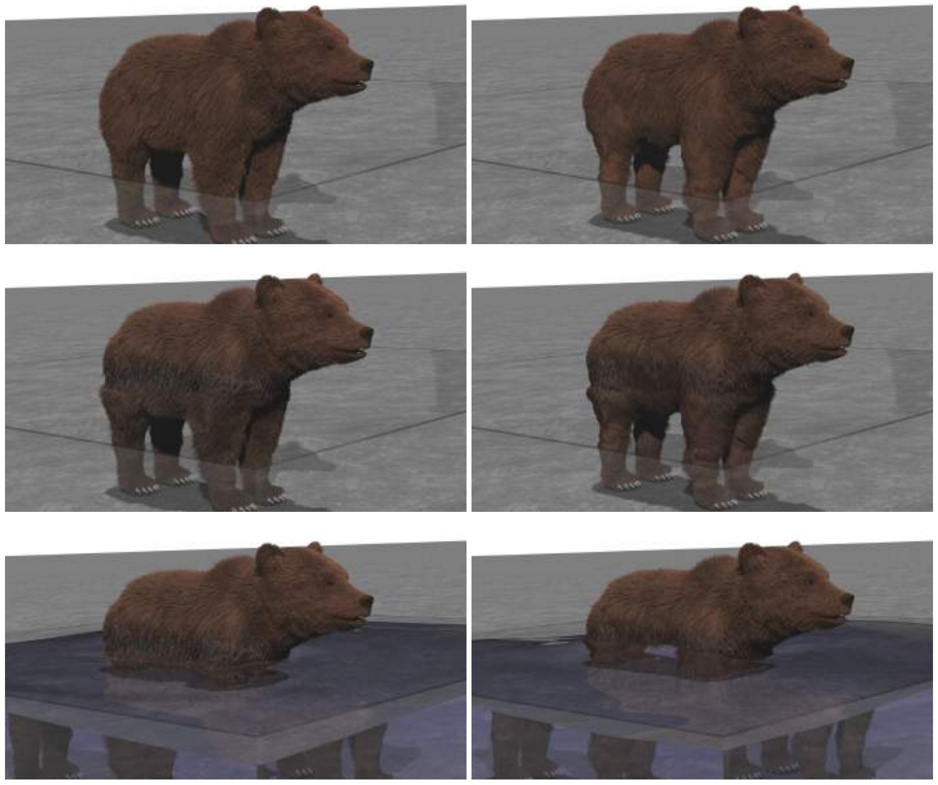


Figure 2.4: Treating hair volumetrically, a simulated dynamic tetrahedral mesh is attached to the KDSM for structure and support, and sways back and forth underwater using the water velocity to apply drag. (Top) No blendshape hair. (Middle) Blendshape hair. (Bottom) Blendshape hair with water.

locations, enabling better collision handling, preserving more hair-hair interactions, etc.

2.3 Layering Framework

Our proposed framework consists of two layers: kinematic skinning layer, and using the kinematically skinnned KDSM to run dynamic simulations and blendshape hairs. The first layer consists of simulating a tetrahedral air mesh, and is described in Section 2.3.1. The second layer involves either dynamic hair simulation (Section 2.3.2) or blendshape hairs (Section 2.3.3), coupled with a hair-water solver as discussed in

Section 2.3.4. Finally, length preservation and pushout are applied as a post-process per frame as discussed in Section 2.3.1.

2.3.1 Skinning Kinematic Hair

Although practitioners often initialize their skinning algorithms using a default or rest pose, and such a process could be used to create our KDSM, we instead independently generate a separate KDSM for each animation sequence of interest in order to obtain higher visual fidelity. Given an animation sequence, we begin on the first frame and construct a tetrahedral mesh for a region of air that will contain the hair for the entire animation sequence. Then the goal becomes to warp this mesh forward in time so that it follows the animation while at the same time capturing other desirable effects. This process is facilitated using the kinematic deformation of the character's skin just as skinning algorithms use the kinematic motions of the character's so-called bones. Moreover, we actually use the kinematic deformation of both the character's skin and its volumetric interior. This is accomplished by repeatedly creating a volumetric morph from the interior of the creature consecutively from one frame to the next using the methods discussed in [7, 30] along with the feature detection from [30] if the mesh topology is not consistent from frame to frame. One could perhaps alternatively use an approach like that of [103] to produce deformations for the tetrahedral mesh to achieve faster simulation rates, given that the reduced model is acceptable.

We generate the tetrahedral mesh for the first frame of an animation sequence using the method of [67] by constructing a level set function whose interior encloses the region of interest. In practice, we experiment with multiple offset values until the level set safely contains all hairs. For examples with long hairs or unusual hairstyles, we create custom geometry and convert it to a level set such that all hairs are safely contained. Using this level set, the method of [67] is used to generate the tetrahedral mesh as usual, and any extra tetrahedra that might be generated too deep inside the creature to be of interest can simply be deleted. A sufficient number of tetrahedral mesh nodes will be required interior to the character in order to guarantee that the tetrahedra adequately cover the desired air region as well as to provide boundary conditions on the mesh that are specified using the results of the volumetric morph. See Figure 2.3 (top), Figure 2.5 (top left), and Figure 2.14 (top right).

Next, the morph is used to move every node of the tetrahedral mesh interior to the creature from frame to frame, while the nodes exterior to the creature can be connected to each other and the interior nodes via one's favorite mass-spring system (as in our implementation) or finite element system and simulated just as in [91, 92] except that the base of our mesh is driven by an underlying morph of the deforming character from frame to frame as opposed to being rigidly attached to a rigid scalp. The interior nodes alone do not provide sufficient boundary conditions because of the relative sparsity of the KDSM particles compared to the desired dense quality of the boundary conditions due to the detailed surface representation and the number of hair follicles. In order to provide sufficient boundary conditions and thus allow for wanted hair shearing, we place additional zero-length springs connecting points embedded on the 2D triangular mesh representing the character's skin to their corresponding barycentric locations in the tetrahedral mesh. It seems that adding these extra zero-length springs wherever one anticipates placing a hair follicle is sufficient. We stress that this dynamic simulation only needs to be done once to skin the air around the creature, and then the results can be stored as a kinematically prescribed motion to be utilized for all future hair animations/simulations that make use of such a motion for their volumetric approximation. As such, it is feasible for an artist to post-process the kinematically deforming tetrahedral mesh using various modeling tools either procedurally or directly as desired. See Figure 2.3 (top row) for an example of a KDSM for an animated creature.

One of the major benefits of this approach is that millions of hairs can be animated just as easily as hundreds since the tetrahedral mesh is indifferent to the number of embedded hairs. With the KDSM defined for the creature, we can now embed hair particles in it. Embedding can be computed via

$$X(i) = \sum_{j=1}^{d} w_{ID(i,j)} Y_{ID(i,j)} \tag{2.1}$$

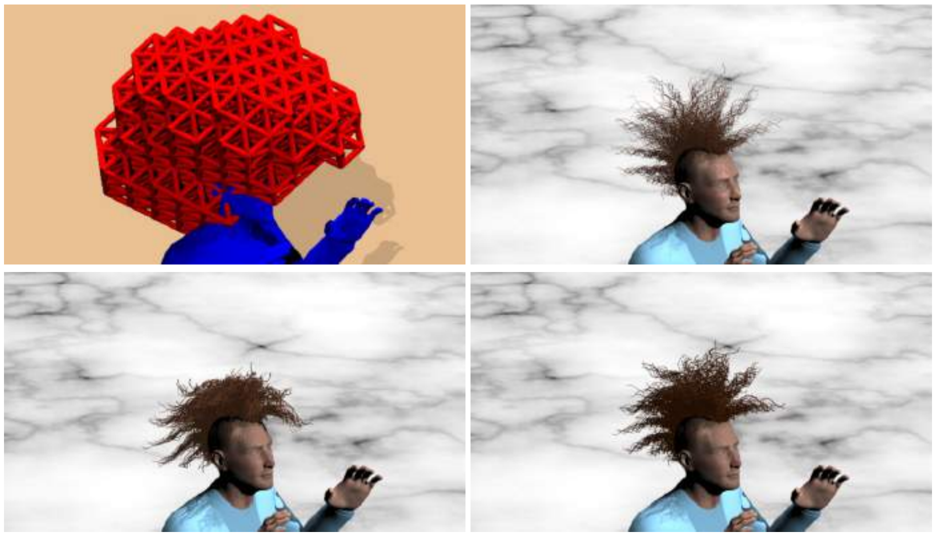


Figure 2.5: (Top left) KDSM. (Top right) No simulation. (Bottom left) Simulation with weak zero-length spring attachments to the KDSM. (Bottom right) Simulation with stronger zero-length spring attachments to the KDSM.

where X is the position/velocity of embedded particle i, Y contains the parent particles' positions/velocities of embedded particle i, and w is a set of barycentric weights. d is the dimension of the simplex from which we are interpolating ($d = 3$ for a triangle mesh and $d = 4$ for a tetrahedral mesh). $ID(x, y)$ returns an index for a parent particle given an embedded particle id x, and an iterator y.

Each hair simply has its base hair particle embedded to follow its corresponding hair follicle on the surface of the character's triangle mesh skin and has all other particles embedded to follow their corresponding locations in the tetrahedral mesh using barycentric weights along the lines of the hard bindings in [82]. As long as the mesh deforms in a reasonable way, the hairs also behave nicely. The only post-processing we do concerns length preservation and collisions with the character, and all hair-hair interactions and self-collision is ignored. Length preservation is accomplished by starting from the root of each hair and shortening each segment to its rest length while modifying the embedded barycentric coordinates of the perturbed particles (similar in spirit to [72, 80]). Interpenetrations with the character are handled using the pushout method from [18] in order to preserve the style of the hair. All

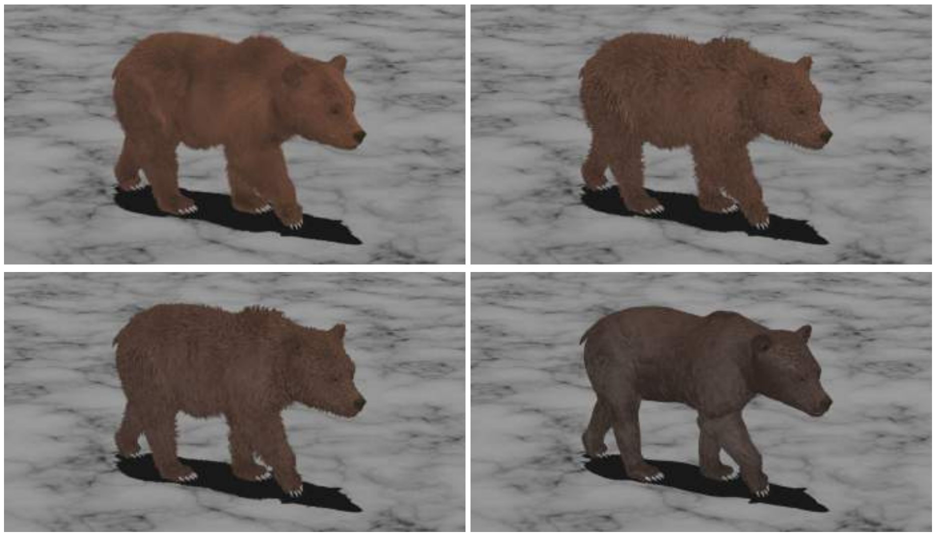


Figure 2.6: Starting from the rest hairstyle (top left), individual hair and volumetric hair simulations as well as procedural techniques were used to create clumped (top right), sagged (bottom left), and matted (bottom right) hairstyles, which were then all bound to the animation sequence from Figure 2.3 using that KDSM.

points that lie inside the character at the end of a frame are projected to a distance outside the character's surface in the range of $[0, \tau]$, where τ is a user-specified value (for $\tau = 0$ all interpenetrating points are projected to the surface of the character, while $\tau > 0$ allows points to maintain some relative offset after projection in order to preserve detail). See Figure 2.3 (bottom row) for an example of hair skinned using this approach.

Typically the hair would only be specified in a reference pose of the character and would not be available on the first frame of an animation. In this scenario, we pre-bake an in-between-animation from the reference pose to the first frame of the animation and apply our method to create a KDSM for this in-between-animation. Of course, an artist may wish to touch-up the resulting hair as desired.

2.3.2 Simulating Dynamic Hair

Dynamic motion can be incorporated into our pipeline on top of the KDSM in one of two ways. The first is to bind a simulated volume mesh to the KDSM and have hairs

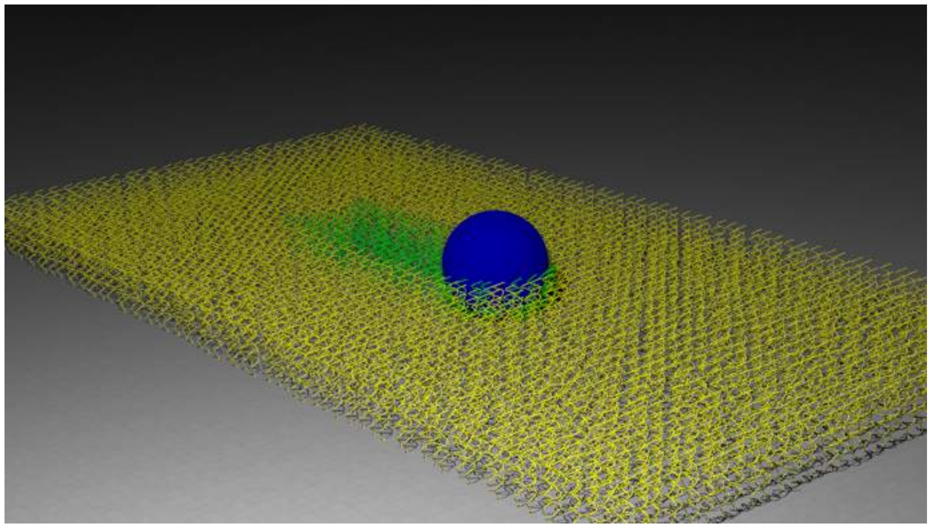


Figure 2.7: Our framework supports selective activation of individual hairs for simulation in order to only activate hairs that are undergoing collision. Here, fully kinematic hairs are shown in yellow while hairs activated for individual simulation are shown in green.

follow embedded positions in this simulated mesh. The second is to activate hairs for per-hair simulation using an underlying constitutive model, and binding these hairs to the KDSM for structure and support.

Simulating a Skinned Mesh

The simplest way to add dynamic motion is to duplicate the KDSM in the first frame, endow this new tetrahedral mesh with one's favorite mass-spring or finite element system, connect it to the KDSM with zero-length springs, and subsequently simulate it. Notably, the strength of the simulated mesh's constitutive model can be reduced significantly from what would otherwise be required because the mesh structure is already supported by its attachment to the KDSM. This provides dramatic increases in efficiency, especially considering that an artist would typically simulate this new mesh over and over with the aim of generating the desired result. Moreover, one can readily refine the simulation mesh where more detail or degrees of freedom are desired and simply bind the refined mesh to the unrefined KDSM using barycentric coordinates without requiring the generation of a new KDSM.

Further efficiencies can be realized by noting that the KDSM already contains a significant portion of the desired motion, e.g. inertial effects and deformation, and thus the task of simulation is merely to generate perturbations to this motion—a task far easier to accomplish! For example, consider the spring force generated by comparing the current length of an edge in the simulated mesh to its rest length in the first frame. Since the endpoint particles tend to follow their paths as dictated by the KDSM via zero-length springs, they will experience compression/expansion forces resisting expansion/compression of the corresponding time-animated edge in the KDSM. That is, the simulated mesh resists the deformation of the KDSM. In contrast, one could use rest lengths that were time-animated to follow their corresponding current lengths in the KDSM, ameliorating the simulated mesh's resistance to the KDSM deformation. Moreover, an artist could choose any intermediate value for the time-animated rest lengths and even vary this choice spatially across the KDSM based on where more or less resistance to the KDSM deformation is desired. This is far more efficient and intuitive than trying to control the strength of springs in order to obtain the desired resistances to deformation in a standard simulation without a KDSM. This strategy can be used to give artists additional control over any geometric-based force using the KDSM as a time-animated guide.

A similar procedure can be used for the time integrals of the force, i.e. the velocity and position. Increments of the velocity or a fraction thereof can be added to simulated particles so that they better follow their corresponding particles in the KDSM, and likewise for position. For non-geometric-based forces such as gravity and drag, since their effects were already included in the KDSM, they do not require inclusion again. However, one might want to use scaled down versions of these forces in order to attain more interesting perturbations of motion for the simulated mesh from the KDSM. Figure 2.4 shows the results obtained by swishing a creature back and forth underwater using the water velocity to apply drag, where the hair motion is obtained by following barycentric coordinates in the simulated mesh.

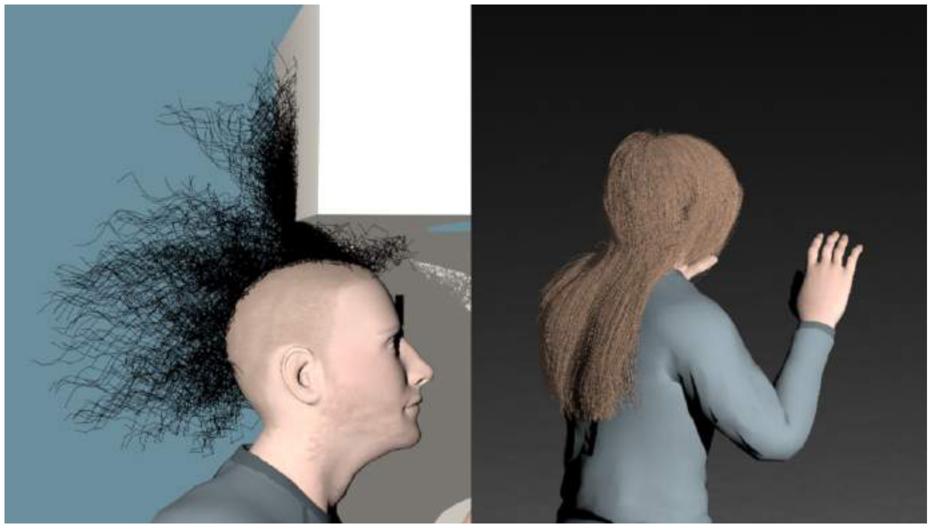


Figure 2.8: (Left) Hairs are individually simulated for high fidelity collision handling. (Right) Individual hair simulation is also used for long hairstyles. Both examples benefit from structure and support provided by the KDSM.

Simulating Individual Hairs

Simulating individual hairs is often necessary when discontinuous behavior between nearby hairs is desired. For example, concerning water simulation, wet hairs clump or cluster due to cohesion/adhesion of the water/hair. A simple way to add dynamic motion to individual hairs is to use soft bindings [82] connecting a zero-length spring between each particle of the hair and its desired position either in the KDSM or in the simulated mesh, whichever is desired. Although this technique works in practice, better results are obtained if the individual hair is given its own constitutive model even if the model is relatively simple using only edge springs using [81]. Although we use the mass-spring method of [81] for hair-hair interactions and collisions for the examples shown in Figure 2.8, we stress that the benefits of the KDSM are agnostic to the underlying constitutive model. Just as was true for the simulated mesh, a relatively weak mass-spring system may be used for added efficiency since the KDSM provides structure and support. That is, any individual hair or guide hair simulation method can benefit from our KDSM. Figure 2.5 shows a mohawk hairstyle undergoing a dynamic simulation, bouncing around and deviating either more or less from the

KDSM based on the strength of the zero-length attachment springs. Figure 2.8 depicts a long hairstyle from [48] that is animated using per-hair simulation with attachment springs to an underlying KDSM. Note that one could achieve hair clumping effects by using adhesion springs between hair curve segments as illustrated in [81], although we did not experiment with this.

Leveraging the fact that the hair will be supported by the KDSM, we utilize a method designed to mostly preserve the shape and style of the individual hair. This is accomplished by creating an individualized tetrahedral simulation structure for each hair, similar to [16], as follows: first we create a bounding box for the hair giving it a characteristic thickness so that it has reasonable structure. Then, we cut this box

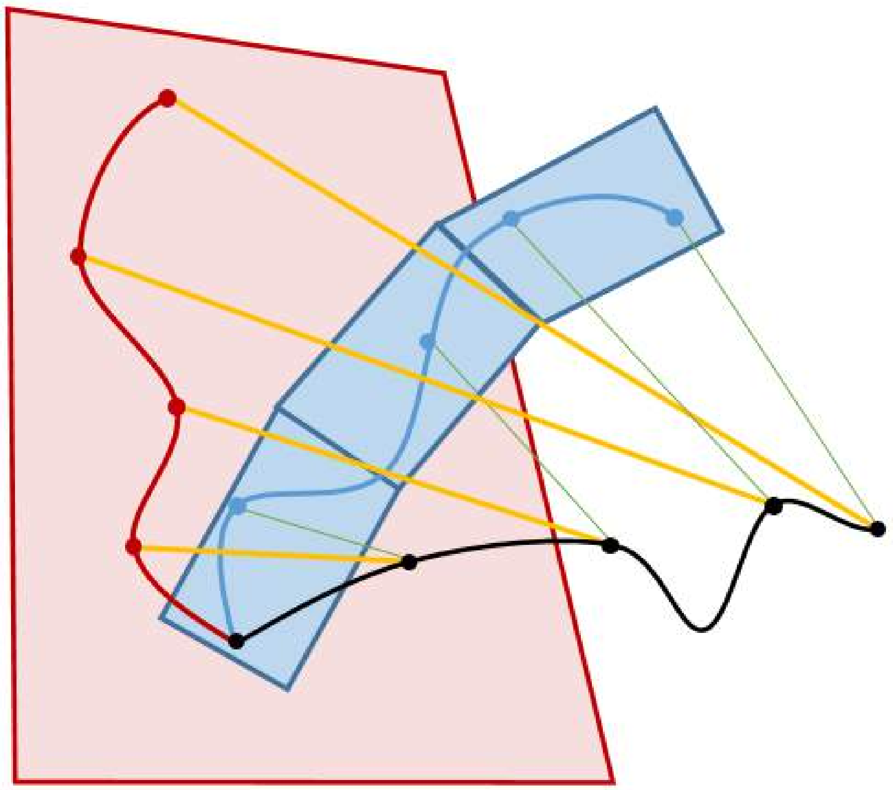


Figure 2.9: Visualization of structure contributing to the final hair shape: the simulated hair (black curve) is connected to embedded positions within the KDSM (red box) and to shape-preserving tetrahedra (blue boxes) by zero-length springs (yellow and green, respectively).

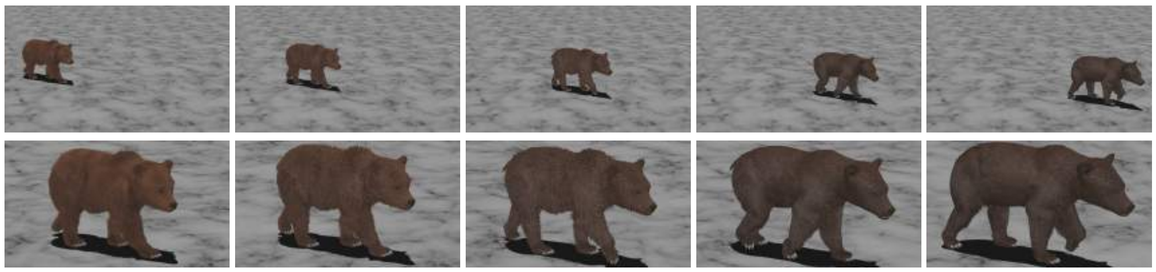
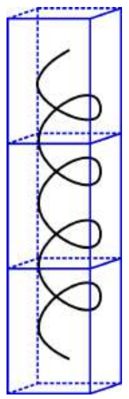


Figure 2.10: (Top row) Using the hairstyles from Figure 2.6 and the KDSM from Figure 2.3 along with our blendshape system, we make a creature grow progressively wetter as it walks. (Bottom row) Close ups of top row.

into 1 to 4 sub-boxes along the length of the hair. Really short hairs get 1 sub-box and regular to long ones get 4, based on trial and error to find the best number.



While a straight hair would tend to go up the center of these boxes, curly hair would weave in a helical pattern through the boxes (see figure). Subsequently, we turn each box into 5 tetrahedra and create a mass-spring model to simulate the entire structure. It is best if the base of the bounding volume lies inside the character or is projected to its surface; either way, it provides Dirichlet boundary conditions following the animation via tetrahedral or triangle embedding respectively. Each simulated hair particle is given a second zero-length spring embedding it into this local tetrahedralized bounding box structure that aids in hair-shape preservation. In summary, each hair particle has two separate connections: one to the KDSM to preserve its structure and one to its tetrahedralized local bounding box to preserve its shape and style, illustrated in Figure 2.9.

In order to further increase the efficiency of individual hair simulation, we often let large numbers of hairs remain unsimulated following the KDSM (or simulated mesh), only activating individual hairs for simulation on an as-needed/desired basis. For example, we can activate individual hairs only when those hairs are in contact or collision with another object similar to [22], as in Figure 2.7, which greatly reduces time spent on simulation discussed in examples section. When a hair is activated, it starts out in its barycentrically bound position and is endowed with its precomputed tetrahedralized local bounding box, internal constitutive model, and constraints. Any individual hair can also be deactivated, at which point each particle of the hair is kinematically returned to its target position in the KDSM (or simulated mesh) slowly under a prescribed time scale using an analytic solution for position and velocity matching (see [98]) so as to not appear jarring. In this way, the user can balance the tradeoff between the fidelity of per-hair simulation and its increased computational

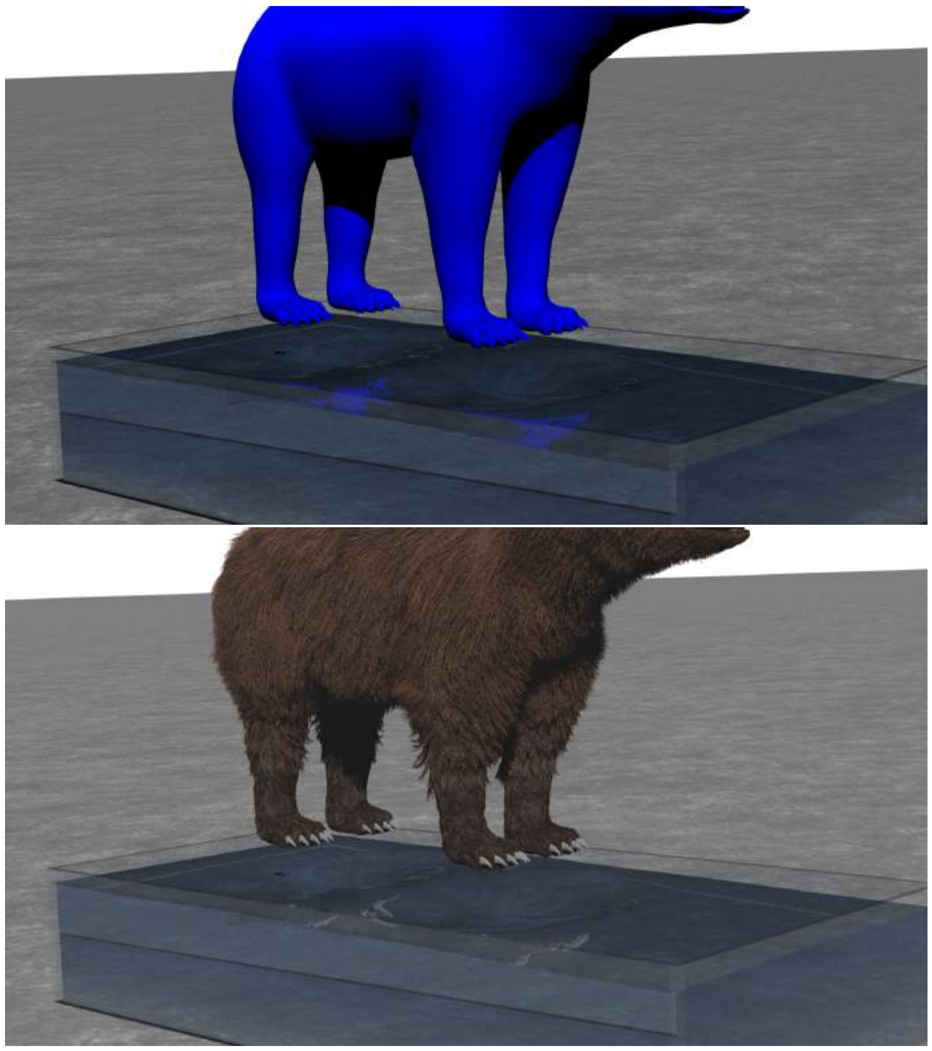


Figure 2.11: (Top) First a water simulation is performed, ignoring the hair. (Bottom) Then in a post-process, a procedurally generated wet map is used in the blendshape system to locally (in space and time) interpolate between various hairstyles such as those in Figure 2.6.

expense with the more efficient approach of driving the hairs through their embedded positions.

Note that the length preservation and object interpenetration strategies discussed in the previous section are also used during dynamic simulation. Also note that at render time one could post-process the hair, blending particle positions between their current simulated positions, their target positions in their simulated local bounding box structures, and their target positions in the KDSM (or simulated mesh). Motivated by [91, 92], we note that a particularly interesting strategy is to use more kinematic motion near the root and more dynamic motion near the tip of each hair.

As an example where the simulation of individual hairs is required, we create a clumped hairstyle by selecting a number of hairs to serve as cluster centers and subsequently using hair-hair attraction forces (see e.g. [21, 24, 94, 95, 81]) to pull surrounding hairs towards the cluster centers. There are obviously a number of ways in which this can be accomplished, but we stress that an important advantage of our method is that we can bind the new hair locations into the KDSM so that any existing animation can readily be played back with either the original hairstyle (see Figure 2.6 top left) or the new clumped hairstyle (see Figure 2.6 top right). That is, as one generates many different hairstyles or perturbations to those styles, they can readily be played back using the kinematic mesh with no additional effort required. One can perturb the clumped hairstyle into a sagged hairstyle simply by simulating a dynamic tetrahedral mesh parented to the KDSM under the effects of gravity and increased mass from water saturation. Figure 2.6 (bottom left) shows the result used in an animation. Similarly, more extreme wetting along with subsequent projection of the hair to the creature's skin results in a matted hairstyle, which can also be pushed through the animation sequence (see Figure 2.6 bottom right).

2.3.3 Blendshape Hair

The KDSM readily facilitates a straightforward implementation of a blendshape hair system. Each hair particle has different barycentric coordinates (potentially in different tetrahedra) for each hairstyle, e.g. Figure 2.6 for dry, clumped, sagged, and

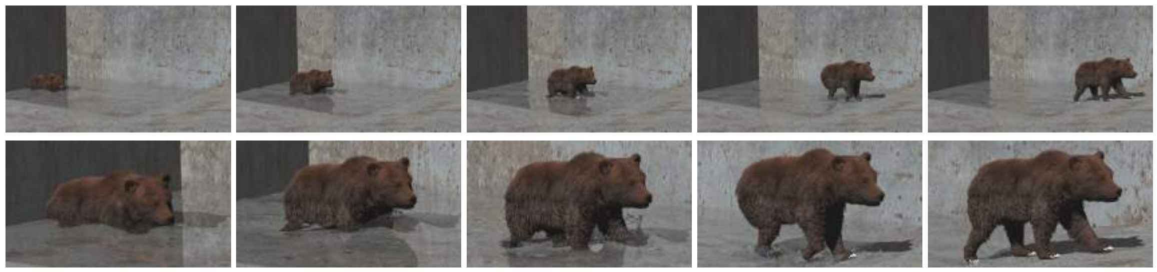


Figure 2.12: (Top) Creature walking out of water. (Bottom) Close ups of top row.

matted hair, respectively. This information can be used to specify the barycentric location of a particle within the KDSM for any combination of hairstyles in standard blendshape fashion. This obviously allows an artist to sculpt new or modify existing hairstyles for use in such a system. Motivated by [21, 94, 95, 93, 45], which describe and model many of the various interactions and styling changes resulting from wet hair, we create a wetness parameter that linearly interpolates from our dry style (see Figure 2.6 top left) to our sagged style (Figure 2.6 bottom left) as the wetness increases. A naive blendshape system could interpolate between between the world space positions of hair particles to target a specific frame of the animation. However, our KDSM automatically pushes this interpolation through the entire animation, allowing us to blend between dry and sagged styles for any subsequent configuration of the creature based on a per-particle wetness. Thus, we can take the KDSM from Figure 2.3 (top) and use our blendshape system to interpolate from the dry style to the sagged style and finally to the matted style, watching the creature grow progressively wetter as it walks. See Figure 2.10. We stress that Figure 2.10 only required dry, sagged, and matted styles in a reference frame, since the KDSM provides a deforming coordinate system which implicitly allows any style in the reference frame to have meaning in every frame of the animation.

A blendshape system provides a dramatic increase in efficiency, especially since obtaining visually impressive interactions between water and hair is often highly subjective and the subject of multiple iterations of artistic and directorial design. A typical implementation of our system would proceed as follows. First, ignoring the hair, run a water simulation to obtain the character's influence on the water taking advantage of the layering of our framework, see Figure 2.11 (top). Then, if desired, simulate a tetrahedral mesh bound to the KDSM, potentially affected by water drag, etc., as in Figure 2.4. Finally, as a post-process, use our blendshape system along with the wetness parameter in order to obtain the desired hairstyle. For example, underwater hair utilizes the dry blendshape while floating around freely following its binding in the dynamic mesh. Hair near the water increases its wetness moving from a dry to a sagged style based on spongelike porous flow effects. Hair that has been pulled out of the water quickly changes from a dry to an either sagged or matted style depending on the degree to which water is retained. See Figures 2.4 and 2.11 (bottom). Additionally, we specify an interpolation weight per particle to control the speed of the interpolation to the desired hairstyle, e.g. so that fur on the creature's paws becomes matted faster than the fur elsewhere on the creature. These weights vary smoothly along the creature's legs so that the blendshape interpolation does not create obvious discontinuities in the hairstyle. When the simulation of individual hairs is desired, this can be done after utilizing the blendshape system by using the blendshape hair to prescribe target locations.

Blendshape systems are notorious for the various artifacts they exhibit. However, our implementation of length preservation and pushout strategies eliminate all visible artifacts from blendshape system. Finally, note that we tuned rendering parameters for each hairstyle and that each hair is rendered using a blending of rendering parameters governed by the same wetness parameter used in the blendshape system to blend the various hairstyles.

2.3.4 KDSM for Hair-Water Interaction

Motivated by [79] (see also the follow-up work of [59]), who proposed the use of an Eulerian Cartesian grid to aid in the simulation of hair-water interactions, we briefly discuss benefits that can be obtained using our KDSM. They rasterize the complex porous structure of hair into a Cartesian grid and subsequently use the porosity values to aid in modeling many interesting wetting effects of hair including water absorption and diffusion, hair-hair cohesion induced by the surrounding water,

water flow throughout the hair volume, water dripping from the hair, and shape transformations of wet hair. Of course, if our water simulations were carried out on an Eulerian Cartesian grid, as is often typical, then their proposed methodology could be implemented on that same grid in conjunction with the water simulation. However, the grid cell size that would be appropriate for a large domain would not have the appropriate resolution near the hair volume, and thus one would have to resort to using a much finer grid near the hair and subsequently coupling the two grids. Note that [79] only focused on the very fine grid very close to the hair. Similarly, our KDSM is also a very fine mesh close to the hair, but it has several distinct advantages over a simple Cartesian bounding volume. For example, it is tighter fitting to the hair volume and thus can more readily be higher resolution. It also moves with the hair, thus once again making it tighter fitting and therefore higher resolution than a Cartesian bounding volume. It also has the potential for higher accuracy since water absorbed by and carried with hair would move along with the KDSM and not require advection, which hinders accuracy. Similarly, water which is diffused through the hair would only need a small motion for the diffusion and could mostly ignore the hair motion since the KDSM and the hair move together. However, water which falls from the hair or moves at a velocity different from the hair would require an arbitrary

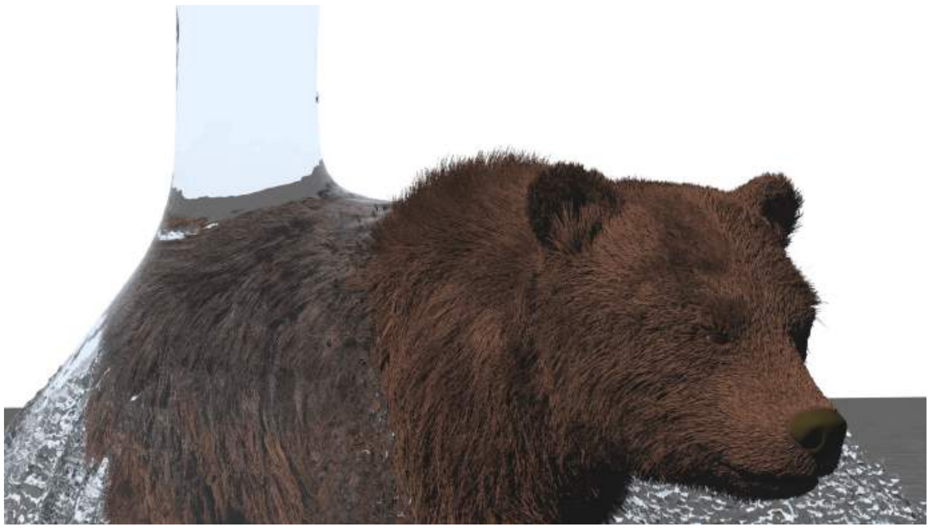


Figure 2.13: Water poured over creature.

Lagrangian-Eulerian style advection scheme as opposed to an Eulerian one.

Along these lines, we briefly illustrate how the KDSM can be used to modify water simulations based on the hair, completing the other half of two way hair-water interaction since we discussed water's influence on hair throughout the paper. One simple but dramatic effect that hair has on water is the drag that hair exerts on it. To accomplish this effect using the KDSM, we first rasterize the hair porosity onto the KDSM using the hair size and occupancy, while also interpolating hair velocity to the nodes of the KDSM. Then it is straightforward to apply drag to any Cartesian grid cell containing water that overlaps any fraction of the KDSM containing hair. Figures 2.12 and 2.13 (see also Figure 2.1) show the KDSM being used in this fashion in conjunction with the rest of the ideas presented in this paper.

Notably, ignoring hair, the KDSM is useful for water interactions with any object as it provides a mechanism for allowing that object to affect the surrounding water in a volumetric way. In Figure 2.14 we create a KDSM for the animation of a creature without hair breaching the water surface. Signed distance values are stored in the KDSM based on the reference pose in which the tetrahedral mesh was created. Then a falloff value based on the signed distance is used to apply drag from the KDSM to the water in a volumetric way using the velocities of the KDSM. This allows one to control the boundary layer effects; in particular, one can control how much water the creature pulls away from the surface simply by turning the drag up and down, as shown in Figure 2.14. This mechanism provides significantly more control to the artist as opposed to relying entirely on the surface of the object to generate these sorts of effects. Note that the artist can also easily control when the water separates from the creature simply by turning down/off the drag.

2.4 Examples

Generating the KDSMs for our examples takes an average of 10 minutes per frame in a single core machine, and frames can be processed in parallel as they have no temporal dependence. Computation time for simulating a dynamic mesh or individual hairs varies, since as the dynamic hair deviates more and more from the original KDSM

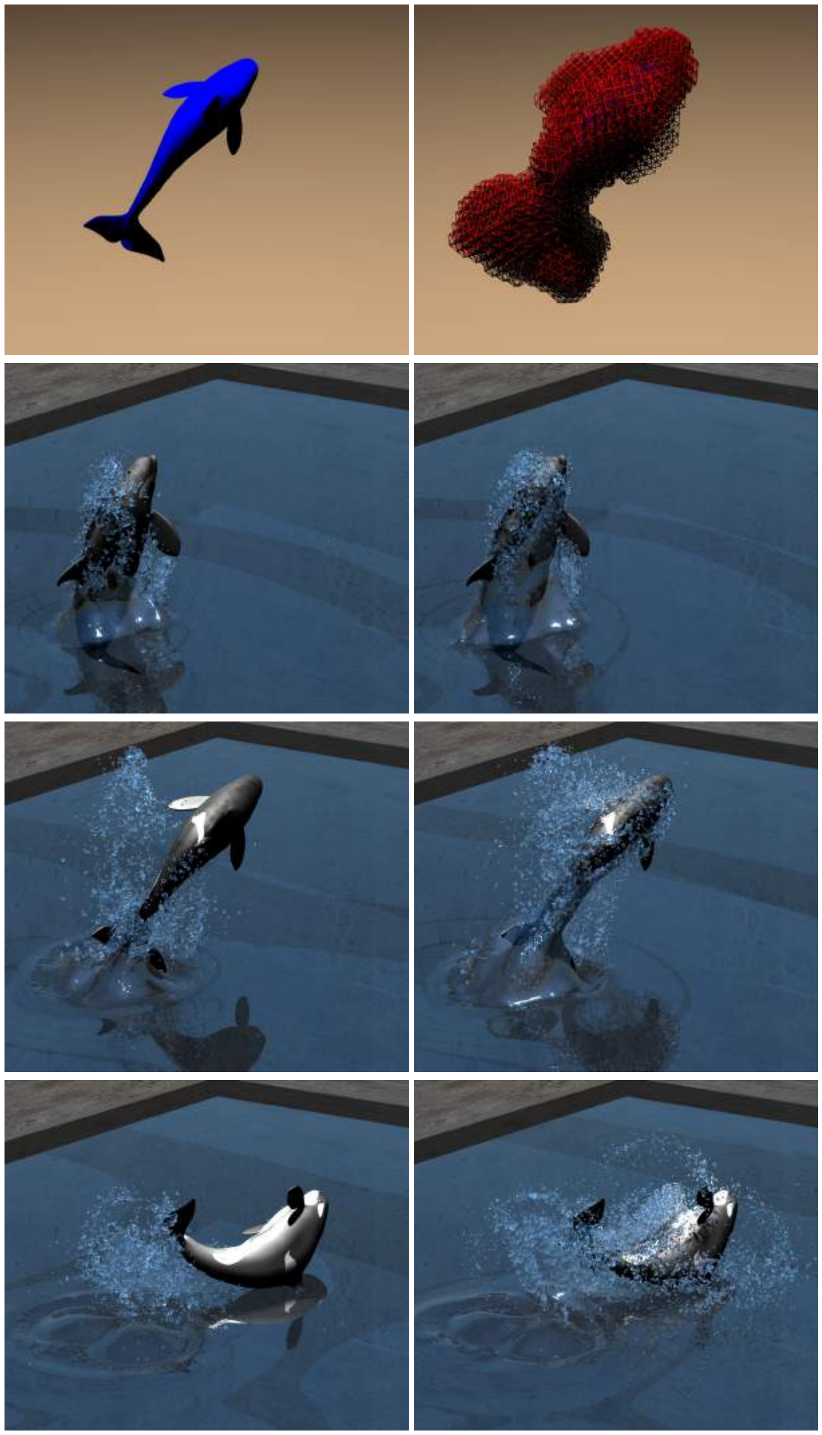


Figure 2.14: (Top) Creature geometry and KDSM. (Left) No KDSM. (Right) Same frames as on the left, except using KDSM for increased drag on the water from the creature.

Example	Hair		KDSM		Frame
	Particles	Strands	Vertices	Tets	Time
Kinematic Mohawk	15,784	1,263	262	895	17s
Dynamic Hair Mohawk	15,784	1,263	262	895	165s
Long Hair Mohawk	69,439	9,703	412	1,557	669s
Carpet	113,693	2,145	8	5	50s
Carpet Adaptive	113,693	2,145	8	5	10s
Kinematic Bear	20,885,989	540,078	3,727	16,170	79s
Blendshape Bear	20,885,989	540,078	3,727	16,170	80s
Bear Slosh	20,885,989	540,078	3,727	16,170	181s
Bear Pour	20,885,989	540,078	3,727	16,170	2183s
Bear Water	20,885,989	540,078	3,727	16,170	2957s
Whale Breach	N/A	N/A	3,661	16,694	201s

Table 2.1: List of examples with their resolutions and average runtime per frame. Note that the first layer, generating the KDSM, is not included.

embedding it will need a stronger mass-spring system in order to sustain its shape, requiring more time to simulate. The blendshape hair component runs in 25 seconds on average on a single core and can be trivially parallelized since the per-hair particle data for the blendshapes is independent of other hairs. We use an altitude spring stiffness of 4.14N and an edge/zero-length spring stiffness of 41.4N for all examples. Dynamic hair which is used to simulate individual hairs uses zero-length springs with stiffnesses of .0414N. The edge spring stiffness of the shape preserving structure is .414N, and its altitude spring stiffness is .0414N. The dynamic KDSM for the bear creature (Figure 2.4 and 2.13) has edge spring stiffnesses of 41.4N and altitude spring stiffnesses of 4.14N. All springs are critically damped. We compare our method to the standard mass spring hair model of [81], which takes 108 hours per frame without self-collisions or repulsions. As expected, our algorithm does not produce the same result because collisions and repulsions are turned off for mass spring hair test due to obvious performance reasons.

The kinematic mohawk example (Figure 2.5) utilizes kinematic skinning only, simply computing barycentric locations of each hair particle on top of the KDSM and applying the length preservation and pushout steps. The dynamic mohawk example (Figure 2.5) utilizes kinematic skinning, the shape preserving hair structure, and soft constraints connecting hairs to both hardbound hairs and shape-preserving hairs. Since individual hairs are simulated, this example takes longer to run than its purely embedded counterpart. The carpet example, which consists of 2,145 hair strands, runs in 50 seconds per frame with all hairs activated, while with adaptive activation the example takes only 10 seconds on average with no more than 150 hairs activated at any time. The kinematic bear example (Figure 2.3) is like the kinematic mohawk example but has significantly more hairs (i.e. 1K hair strands vs 540K hair strands). The blendshape bear example (Figure 2.10) is running blendshape hair on the results of the kinematic bear example. There is virtually no difference in timing for this example compared to the kinematic bear since we load all blendshape data from the disk at the very beginning of the simulation and simply run linear interpolation on each hair particles while the kinematic skinning test has to compute each hair particle's location every frame. The bear slosh and bear pour examples (Figure 2.4 and 2.13) use dynamic skinning. The bear water example (Figure 2.12) is run with particle level set water simulation, porosity rasterization from the KDSM, and blendshape hair. Note that the performance of bear examples involving hair-water interaction is dominated by water simulation times; hair components maintain similar performance compared to examples without hair-water interaction. The whale breach example (Figure 2.14) consists of a particle level set water simulation with adhesion applied to the background eulerian grid. Examples were run on a desktop machine with a 12 core 3.06GHz CPU, 96GB RAM, and a 500GB SSD. A summary of the resolutions and timings of our examples is shown in Table 2.1.

2.5 Conclusion

We proposed a new KDSM data structure that allows one to skin the motion and deformation of millions of hairs, and showed how this data structure greatly increases the efficiency of subsequent simulations whether or not the hair is treated as a volume or as individual hairs. We also showed that the KDSM enables a quite simple implementation of a blendshape hair system. Notably, the KDSM is intrinsically part of the animation and does not dictate any particular approach to simulation. Thus, one can use their favorite techniques, whether simulating hairs individually or as a volume, whether using masses and springs or finite elements or any other model, etc. A KDSM provides structure and support greatly increasing the efficiency of any simulation method. Furthermore, we illustrated that the KDSM can be quite useful for hair-water interactions and even for water-character interactions without hair. As future work, we plan to investigate how one might create a volume of fluid solver via an arbitrary Lagrangian-Eulerian framework directly on the KDSM and subsequently couple such a solver to a standard water simulation on a background Cartesian grid—albeit noting that it can often be quite difficult to render the results of a volume of fluid simulation in a visually pleasing manner. It would also be interesting to consider using our approach for cloth simulation instead of hair.

Chapter 3

Character-Water Interaction

In this chapter, we propose a novel volume conserving framework for character-water interaction, using a novel volume-of-fluid solver on a skinned tetrahedral mesh, enabling the high degree of the spatial adaptivity in order to capture thin films and hair-water interactions. For efficiency, the bulk of the fluid volume is simulated with a standard Eulerian solver which is two way coupled to our skinned arbitrary Lagrangian-Eulerian mesh using a fast, robust, and straightforward to implement partitioned approach. This allows for a specialized and efficient treatment of the volume-of-fluid solver, since it is only required in a subset of the domain. The combination of conservation of fluid volume and a kinematically deforming skinned mesh allows us to robustly implement interesting effects such as adhesion, and anisotropic porosity. We illustrate the efficacy of our method by simulating various water effects with solid objects and animated characters with our novel surface reconstruction method.

3.1 Introduction

Character-water interaction is a widespread phenomenon in the visual effects industry, and there have been many efforts to push for higher quality water interaction with animated characters such as King Kong in *Kong: Skull Island (2017)*, Hank the octopus in *Finding Dory (2016)*, and various characters in *Moana (2016)*.

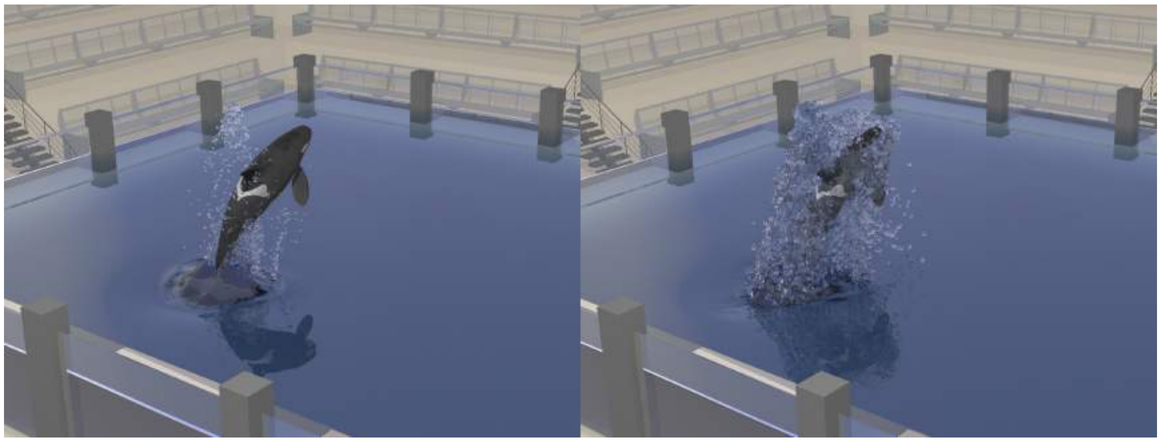


Figure 3.1: (Left) Whale breaching with the PLS method using automatically generated removed particles for spray. Very little of the water volume follows the whale's motion because of volume loss on the relatively coarse Eulerian background grid. (Right) Using the same Eulerian grid, our ALE based VOF method on the KDSM produces much more visually interesting sheeting and spray effects.

Arguably, the most obvious approach to obtaining more detailed features anywhere in the domain is to place more degrees of freedom in the region of interest. A number of adaptive methods have been developed such as Adaptive Mesh Refinement (AMR) [89], octree data structures [61], [60], [1], lattice based tetrahedral methods [25], [11], [8], and Chimera grids [33], [34]. While these methods greatly improve water simulation detail through adaptivity, various authors have noted numerous drawbacks including the need to remesh very often, difficulties in implementation, performance bottlenecks induced by high communication costs, and issues related to domain decomposition due to a large number of small patches. These issues are exacerbated when the adaptivity is required near boundaries with animated characters, since the character motion can rapidly change the region in space where the adaptivity is required. A more natural approach would be to use an adaptive mesh that moves with the character such as the recently proposed kinematically deforming skinned mesh (KDSM) of [56]. This allows one to prebake the adaptivity so that on-the-fly refinement is not required during the simulation. This makes the method straightforward to implement and robust in its handling of delicate phenomena.

Even with additional degrees of freedom near the animated character, the highly

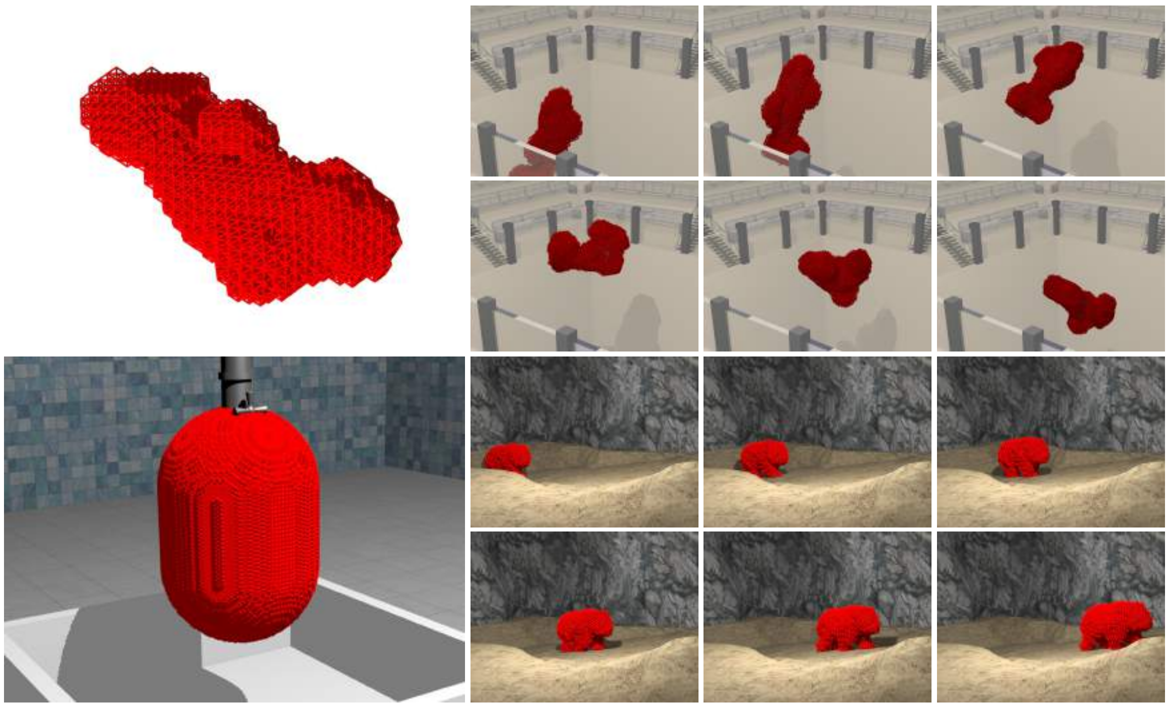


Figure 3.2: (Top Left) A KDSM mesh around a whale in a normalized pose (also known as T-pose or rest pose). (Top Right) A sample animation showing the KDSM skinned to follow an animation of a whale breaching. (Bottom Left) A KDSM mesh around the ball. (Bottom Right) A sample animation showing the KDSM skinned to follow an animation of a bear walking on a shore.

dynamic water motion and thin films are notoriously difficult to simulate due in large part to both volume loss and difficulties with imposing proper boundary conditions between the water and the character. We address volume conservation by proposing a novel volume-of-fluid (VOF) method implemented on the KDSM. Although our proposed VOF method is novel, it is similar in spirit to other VOF methods such as [65], [66] in that no fluid volume is lost, especially as compared to typical Eulerian methods. VOF method is a well known technique as demonstrated in [47], [17], [77], and [90]. There have been some recent interesting works on boundary conditions between solids and fluids such as [108], [105] using an Eulerian fluid grid (see [6] for SPH); however, it is more natural to specify these types of boundary conditions when the fluid grid is moving along with the solid in its Lagrangian frame, even if it is

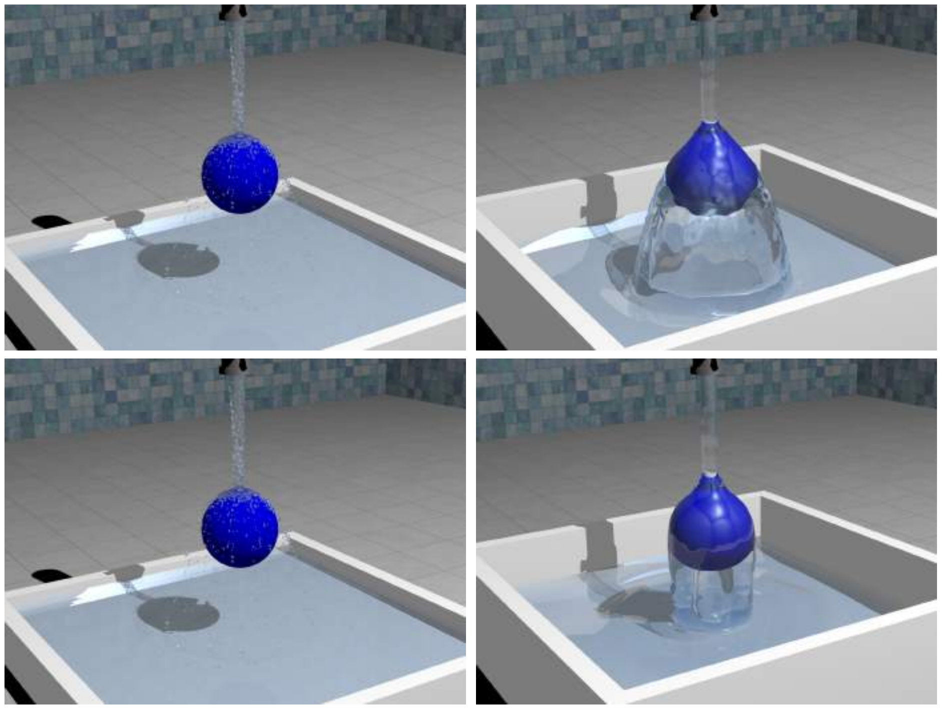


Figure 3.3: Same as Figure 3.4, but using an even smaller water stream accentuating the benefits of our approach especially when considering volume conservation.

deforming a bit in that frame as is the case with KDSM. With this treatment, much of the fluid moves along with the mesh being driven by the character animation (which is also driving the mesh) meaning that less fluid volume flows from one computational cell to another. This is the typical arbitrary Lagrangian-Eulerian (ALE) approach, see for example [39], [40], [55]. Notably, our method significantly differs from existing ALE implementations in that our ALE mesh is prebaked based on kinematically prescribed motion and has topology that remains consistent throughout the entire animation sequence. This separation of the remeshing step resolves a key problem of ALE based methods which can lack robustness due to the numerical instabilities caused by ill-formed elements–this can now be addressed during a preprocessing step.

In order to increase the overall efficiency and efficacy of our approach, we only utilize the ALE based VOF method on the KDSM near the animated character while using a standard Eulerian based Cartesian grid solver in the rest of the domain, in our case the particle level set (PLS) method [35], [36], including spray particles [62]. It is important to note that the PLS method is a hybrid method combining particle and grid representations, and early work was presented in [43] where they implemented a precursor to PIC/FLIP while removing unneeded interior particles far from the surface to boost performance and using level set to reconstruct smooth surface. Recently, [41] proposed further improvements of PLS and FLIP hybrid method. Thus, PLS and PIC/FLIP share important commonalities, and our VOF method can improve PLS or PIC/FLIP or any other method combining particle and grid representations.

Note that our spray particles carry mass and momentum as in [62] for visual quality when spray particles interacts with water surface instead of using massless particles as in vanilla PLS. Importantly, the conservative nature of our VOF solver allows for relaxation of the numerical approach especially since the VOF solver is only required in a small subset of the domain near the character while a standard Eulerian solver is used elsewhere. Thus, we devise a straightforward partitioned (as compared to monolithic) approach to the coupling of the fluid flow equations between the Eulerian Cartesian grid and the ALE based VOF solver on the KDSM; see Section 3.4. Our partitioned approach allows unit testing for each component which greatly streamlines the development process. Importantly, this combination of a standard Eulerian solver on the bulk of the domain with an ALE based KDSM mesh near the character allows the proposed VOF scheme to be incredibly simple, as discussed in Section 3.3, which is quite notable given the typical high level of complexity one usually confronts with VOF methods.

The first contribution of this paper is our strategy of prebaking the dense ALE mesh which occupies the space near the object or creature of interest, taking advantage of the adaptivity to capture detailed water phenomena based on the intuition that most interesting water effects are focused near the creature. We achieve a robust simulation method by separating the nontrivial mesh processing operations from the simulation stage and incorporating them into a preprocessing stage, where we precompute various auxiliary data in order to improve the performance of the simulation. Our second contribution is our novel VOF method, which conserves volume within the ALE mesh, whereas the PLS method in the background alone does not. Our approach of conserving volume near the object or creature of interest allows us to

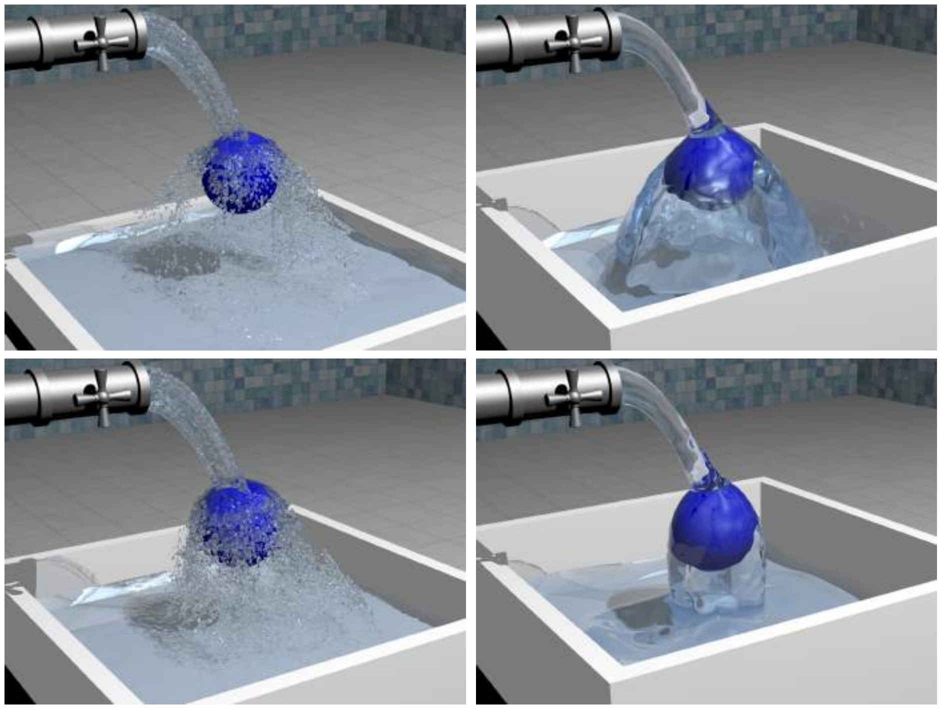


Figure 3.4: (Top Left) A standard PLS simulation on a relatively coarse Eulerian grid. (Top Right) Our ALE based VOF method on the KDSM achieves better water sheeting and volume conservation using the same Eulerian grid. (Bottom Row) Applying adhesion forces to both simulations produces the desired clinging to the ball with our method but has almost no effect on the PLS simulation.

implement various adhesion and porosity effects robustly and with mechanisms for artistic control. The third contribution of our method is the straightforward partitioned approach for coupling the coarse background Eulerian grid and our fine ALE mesh, which greatly streamlines the development process. Last, we demonstrate our novel water reconstruction scheme to capture smooth, time-coherent water surface from our VOF water merged with the background water.

3.2 KDSM

Following [56], we generate a KDSM from a tetrahedral BCC (body centered cubic) lattice as in [67] using a thickened level set of the triangulated surface skin mesh of a creature or an object in a normalized pose (see Figure 3.2 left). Then, given an animation sequence of the creature's triangulated surface skin mesh, the KDSM nodes inside the creature are morphed to follow the animation as per [7, 30] capturing the kinematic deformation of the creature's skin and its volumetric interior. We connect the KDSM nodes that are exterior to the skin mesh of the creature to one another and to the internal nodes via a constitutive model (e.g. mass spring), so that the KDSM nodes that are external to the creature also follow the animation (see Figure 3.2 right). In addition, zero length spring attachments are connected between the creature's skin mesh and corresponding barycentric locations in the KDSM in order to obtain more accurate deformations of the KDSM near the creature's skin.

[56] embedded hair particles in the KDSM so that they would follow the skinned animation sequence. Each hair's base particle is embedded on the surface of the character's triangle mesh skin and the rest of hair particles are embedded in the tetrahedral mesh using hard bindings as in [82]. They also showed how a duplicated KDSM which is following the original constrained with zero length springs could be used to produce more dynamic hair behavior. Since the original KDSM sequence contains a significant portion of the desired motion, further iterations for dynamic hair behavior becomes much more efficient. In addition, they showed the benefits of the KDSM when simulating individual hairs as well as how to use the KDSM to implement a blendshape system including the effects of clumping, sagging, and matting. They simulated individual hairs using [81] and soft bindings (see [82]) connecting hair particles and their corresponding desired positions in the KDSM via zero length springs. They proposed a shape-preserving tetrahedral column to maintain the original style of the hair also connected to hair particles via zero length springs. The blendshape hair component is implemented as a collection of barycentric coordinates for each hair particle, and can be interpolated in time to implement a change in hairstyle over time (e.g. bear getting wet in the water). Then length preservation and interpenetration is run as a post-process, shortening the each hair segment to its rest length (see [72, 80]) and applying pushout method from [18]. Interestingly, they demonstrated how the air volume enclosed by the KDSM could be utilized to add adhesion and drag effects, porosity for hair, etc.

A major shortcoming of this prior work in regards to water simulation is that the KDSM is merely used to provide information such as forces that augment the treatment of the Eulerian grid. Thus, all the typical drawbacks of volume loss, etc., are not only still present but potentially worsened by these additional forces. See, for example, Figure 3.4 and 3.3.

3.3 KDSM Fluids

Our novel ALE based VOF method on the KDSM produces compelling results even though it has none of the complexities associated with typical VOF methods–even the volume conservation step is quite simple. We stress that the ability to use such a simple method is due in large part to the partitioned coupling discussed in Section 3.4, the adaptivity of the KDSM, exact volume conservation, and the Lagrangian nature of the KDSM as it follows the animated creature. Our VOF method fully conserves volume within the KDSM, while the rest of the domain simulated using the PLS method does not.

3.3.1 Precomputation

Starting from the original KDSM animation, we precompute auxiliary information such as adaptivity by subdividing the KDSM until a desired resolution is reached (for our examples, we subdivided KDSM multiple times until the size of tetrahedron is 4 to 8 times smaller than the background grid cell size). Prebaking subdivisions to obtain per-frame ALE meshes with consistent topology greatly increases the robustness and minimizes computation time as compared to typical ALE remeshing. We subdivide using [67], but only utilized the subdivision operation part without any adaptivity features as we wanted an evenly subdivided ALE mesh. Although we opt to precompute our KDSM, one could subdivide on-the-fly if desired. Note that we use a relatively coarse KDSM to run a mass spring simulation and then subdivide the resulting KDSM for performance reasons. One can always use a dense KDSM to obtain better boundaries near the creature's skin, but we found the current scheme sufficient for our examples. We additionally precompute a number of useful quantities such as per node velocities and a level set volume for every frame of the animation. Every cut cell tetrahedron, those containing a part of the creature's surface, is assigned a surface normal and object velocity. Those surface normals and object velocities are extrapolated to every tetrahedron of the KDSM exterior to the creature using the level set information.

Per tetrahedron solid occupancy is precomputed in the cut cells using point samples and the quadrature formula of [107] testing how many point samples are inside of the creature using the level set representation. Instead of using the exact volume, we simply use the fraction of point samples inside and outside the creature to compute an approximate volume. This added simplicity is equivalent to a slight sub-tetrahedral perturbation of the solid surface. Obviously, this could be done more accurately, but we found that this simple method worked quite well, and simplicity and efficiency is highly desirable when one might want to refine near the surface of the creature on-the-fly.

Our proposed volume conservation method is motivated by the shock propagation for rigid bodies from [44]. As such, we precompute the rank of each tetrahedron as its topological distance from the creature, similar in spirit to the contact graph from [44]. Tetrahedra fully inside the creature are assigned a rank of -1, and partially filled tetrahedra are assigned rank 0. Then, all tetrahedra with unassigned ranks that are node neighbors of rank 0 tetrahedra are assigned rank 1. Rank 2, rank 3, etc. are assigned similarly.

3.3.2 Advection

We store vector valued velocities as per tetrahedron values, and multiplying by the mass of water in a tetrahedron yields momentum. Our ALE based VOF method updates the velocity and the amount of water in each tetrahedron on a new mesh given values on an old mesh.

First, for each tetrahedron, we trace its nodes backwards in time using its vector valued velocity multiplied by Δt. As shown in Figure 3.5, this rigidly translates the

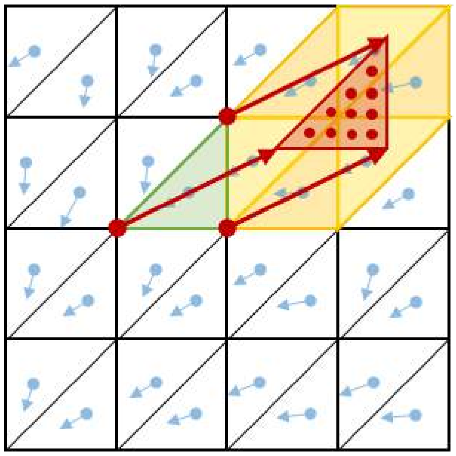


Figure 3.5: A backtraced triangle shown in red uses each of its 10 quadrature point samples to transport water from the old mesh to the new one. Here, we render both the old and the new mesh in the same location assuming that the KDSM is not moving for simplicity of depiction. Each red dot point sample would attempt to remove 1/10 of the area of the red triangle from the specific yellow triangle that it is interior to.

tetrahedron. Alternatively, one could instead compute per node velocities, but we have found this unnecessary. If a backtraced node collides with a solid surface, it is clamped to that location. Thus, the backtraced tetrahedron does not deform unless it hits a solid surface. The resulting backtraced tetrahedron (shown in red in Figure 3.5) is used to collect water from the old mesh in order to deposit it on the new mesh. Instead of performing the usual complex geometric intersections between backtraced tetrahedra and the old mesh, we take a simpler approach using a number of quadrature formula point samples (again from [107]). Each point sample (shown as a red dot in Figure 3.5) attempts to transport a certain amount of water from the old mesh tetrahedron it lies within (shown in yellow in Figure 3.5) to the original tetrahedron on the new mesh (shown in green in Figure 3.5). This potential amount of transported water is calculated as the volume of the backtraced tetrahedron divided by its number of point samples. Both water volume and associated momentum are transported. If a point sample falls outside the KDSM, we compute the amount and momentum of water to transport using interpolation from the background Eulerian grid. Note that this water is not removed from the background grid, since the background grid is treated in an Eulerian fashion and updated properly via the coupling proposed in Section 3.4.

The aforementioned advection might attempt to transport more water out of a tetrahedron on the old mesh than that tetrahedron contains. This could occur because of size differences between tetrahedra or because multiple point samples from separate tetrahedra of the new mesh request water from the same tetrahedron on the old mesh. Thus, in a second step, we visit every tetrahedron on the old mesh and scale down the amount of water each point sample removes in order to match the total volume of water in this old mesh tetrahedron as in [57].

In a third step, we identify any tetrahedra on the old mesh that may have excess water, which was not transported to the new mesh, and forward advect this water to the new mesh similar to [57], [58], [55]. However, our method can be much simpler because we track the exact volume of water with our VOF method, and therefore do not mind overfilling tetrahedra because they are drained to their appropriate volume during the volume conservation step (see Section 3.3.3). For each tetrahedron on the old mesh that requires forward advection, we use that tetrahedron's velocity to advect its nodes forward in time (in the opposite direction of Figure 3.5) and use its point samples to locate which tetrahedra in the new mesh will receive its water. Similar to backward advection, we clamp nodes that collide with solids, use the number of point samples and the original tetrahedron volume to decide on the fraction of water deposited in each target tetrahedron, and utilize special treatment for any point sample that leaves the KDSM and lands on the background Eulerian grid. In particular, we find particle creation to be quite useful for transporting water off of the KDSM (see Section 3.4.1).

3.3.3 Volume Conservation

Our volume conservation method is used to enforce incompressibility on the new mesh using our precomputed rank. The method consists of three parts: smear, pushout, and velocity correction. Pushout transports excess water and associated momentum outward from the creature's skin mesh using rank motivated by [44]. However, this

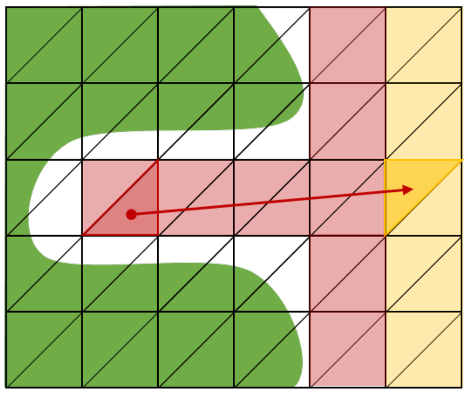


Figure 3.6: The triangles cut by the green solid surface region would be assigned rank 0, their node neighbors shown in red would be rank 1, and their node neighbors shown in yellow would be rank 2. The arrow shows how a rank 1 triangle needs to look non-locally in order to find a rank 2 triangle that it can deposit excess water into.

ignores the fact that the fluid can rotate and move laterally, so we first apply what we refer to as a smearing step to account for this behavior. Both the smear and the pushout step transport the volume and associated momentum together. Finally, velocity correction is used to apply boundary conditions on the water from the creature.

We refer to tetrahedra as oversaturated when they contain more water than their volume should allow. The smearing step loops over oversaturated tetrahedra, except those on the boundary of the KDSM, and distributes the excess fluid equally to face neighbors. The boundary tetrahedra are taken care of in the pushout phase.

For pushout, we iterate over oversaturated tetrahedra in order from the lowest rank to the highest starting with tetrahedra of rank 0 which intersect the creature's skin mesh. For each oversaturated tetrahedron, we distribute as much excess fluid as possible equally to its face neighbors that are not yet fully saturated. If there is still excess fluid after every neighbor is saturated, it is distributed equally to all face neighbors with strictly higher ranks. Note that it is possible to have a tetrahedron without any valid neighbors to distribute water to, since the creature skin mesh can have narrow space between solid surfaces as illustrated in Figure 3.6. To handle this case, we preprocess the neighbor information using breadth first search to look for non-local neighbors with higher ranks to properly transport excess water to. Although

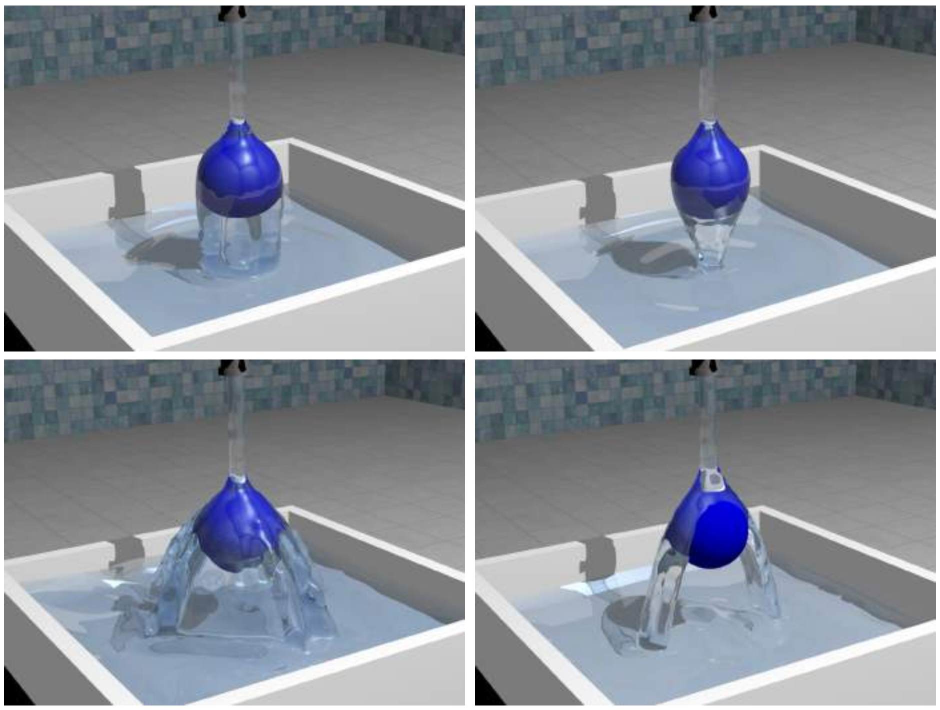


Figure 3.7: Our ALE based VOF method provides robust adhesion control to produce various effects. All four of these examples are obtained by merely varying adhesion effects.

not trivial, this can be done in the preprocessing step after determining rank. For boundary tetrahedra, we transfer excess fluid to the background Eulerian grid for particles as discussed in Section 3.4 and 3.4.1, respectively.

Finally, velocity correction is used to apply boundary conditions on the fluid from the creature. We assign a Boolean flag per tetrahedron indicating whether its fluid velocity needs to be corrected, and initialize all flags to false. Then, we iterate over all cut cell tetrahedra with water and set flags to true. Subsequently, we loop over the tetrahedra in the same order as in the pushout phase, clamping the normal component of the fluid velocity to be the precomputed object normal velocity when the flag is set to true and the normal component of velocity is smaller than the precomputed object normal velocity. Higher rank tetrahedra have their Boolean flag set to be true when they have a lower rank face neighbor which is fully saturated that required clamping. This limits the enforcement of boundary conditions to those tetrahedra exposed to the creature surface by a column of water. Although this removes circulation on the KDSM, the circulation is restored from the background Eulerian grid during coupling as discussed in Section 3.4.

This approach is not a standard projection scheme, but still enforces a divergence free condition thereby enforcing incompressiblity. One other notable approach that is not an advection-projection scheme is [106]. To elaborate, fluid simulated using the Navier-Stokes equations is assumed to obey the conservation of mass equation $\partial\rho/\partial t + \nabla \cdot (\rho\mathbf{u}) = 0$ (i.e. no fluid is created or destroyed). Here, $\mathbf{u}$ is the fluid velocity, t is time, and ρ is the fluid density. By the product rule, this is equivalent to $\partial\rho/\partial t + \rho\nabla\cdot\mathbf{u} + \mathbf{u}\cdot\nabla\rho = 0$. Using this equation, it can be seen that setting $\nabla\cdot\mathbf{u} = 0$ is equivalent to setting $\partial\rho/\partial t + \mathbf{u}\cdot\nabla\rho = 0$. Either condition implies the other–they are equivalent from the conservation of mass. Setting $\rho = m/V$ and using the product rule gives $(1/V)(\partial m/\partial t) - (m/V^2)(\partial V/\partial t) + \mathbf{u}\cdot((1/V)\nabla m - (m/V^2)\nabla V) = 0$, which can be regrouped as $(1/V)(\partial m/\partial t + \mathbf{u}\cdot\nabla m) - (m/V^2)(\partial V/\partial t + \mathbf{u}\cdot\nabla V) = 0$. The first term $(1/V)(\partial m/\partial t + \mathbf{u}\cdot\nabla m)$ must equal zero since mass is conserved along streamlines. This means $\partial V/\partial t + \mathbf{u}\cdot\nabla V$ must also equal zero, and taking this with the divergence free condition $\nabla\cdot\mathbf{u} = 0$ yields $\partial V/\partial t + \nabla\cdot(\mathbf{u}V) = 0$, i.e. conservation of volume. Thus, conserving volume is equivalent to enforcing a divergence-free velocity

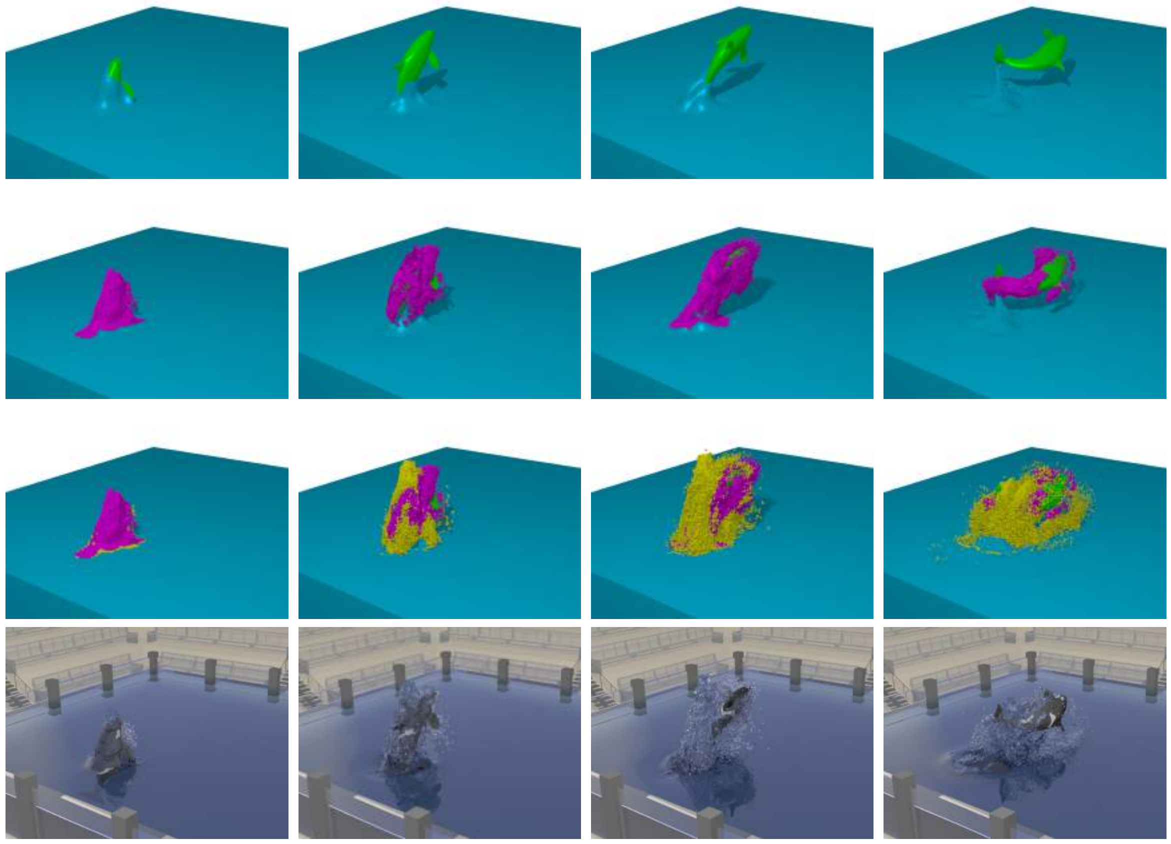


Figure 3.8: A whale breaches out of the water. (Top Row) Visualization of the Eulerian water. (Second Row) VOF tetrahedra water is rendered in pink. (Third Row) Particles are shown in yellow. (Fourth Row) Final rendering.

field.

With regard to maintaining the incompressibility of a simulated fluid, we note that a standard projection scheme such as the classic method introduced by [29] is just one proposed algorithm for this task. In fact, Chorin advocated at least two distinct schemes [28], though the advection-projection procedure became most popular. Chorin-style projection claims that one can advect a fluid state ad-hoc off of the manifold of all incompressible fluid fields and then correct the ensuing error by projecting back onto that manifold. However, these projection-style schemes are known to be brittle (e.g. they require small time steps and do not even always converge under temporal refinement), and they are well-known to be unable to capture important physics of fluids (such as viscosity, due to its parabolic nature). Thus, solving a pressure Poisson equation to enforce a divergence-free velocity field is not the ultimate, and certainly not the only, numerical scheme to simulate incompressible flow. Our technique, which indeed differs from a pressure projection, is able to successfully and robustly conserve volume, which as shown above implies the standard incompressibility condition. Hence our volume conservation method maintains a faithful relationship with the underlying fluid principles and equations. Moreover, at a high level, we remark that even the Navier-Stokes equations quickly fail to be a physically accurate model when considering real-world flow problems i.e. turbulence; however, such approximations are heavily relied upon in computer graphics due to their computational amenability and their ability to produce visually plausible results.

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 [8]. 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 Figure 3.9. In Figure 3.9, 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.

3.3.4 Adhesion

We allow an artist to paint adhesion coefficients α and force directions $\vec{d}$ on the triangulated surface mesh of the creature, and then we rasterize this information to the KDSM setting adhesion quantities in rank 0 tetrahedra. We propagate adhesion quantities to face neighbors (or non-local neighbors) of strictly higher rank tetrahedra by averaging the adhesion quantities from lower rank neighbors for which adhesion had already been specified. We apply an adhesion force $\alpha\vec{d}$ when a tetrahedron is within a prescribed distance ϕ_a from the creature's surface with linear falloff, i.e. $\alpha(\phi_a - \phi)/\phi_a\vec{d}$ where ϕ is the distance from the creature's surface (similar to [109]).

Figure 3.7 illustrates some of the many interesting visual effects obtainable by varying adhesion parameters. Notably, it is the robust volume conservation of our VOF method and the adaptivity of the KDSM that allows for such interesting effects. We attempted similar simulations using a standard Eulerian method and mostly achieved disturbing volume loss. The top row shows how increasing adhesion (from left to right) makes the water to stick to the ball and flow around to the bottom surface before separating. The bottom left image was created with vectors $\vec{d}$ pointing outwards from the ball at various locations to produce thickened streams. In contrast, the bottom right figure shows how the vectors $\vec{d}$ can be used to direct water away from parts of the ball's surface, drying it out.

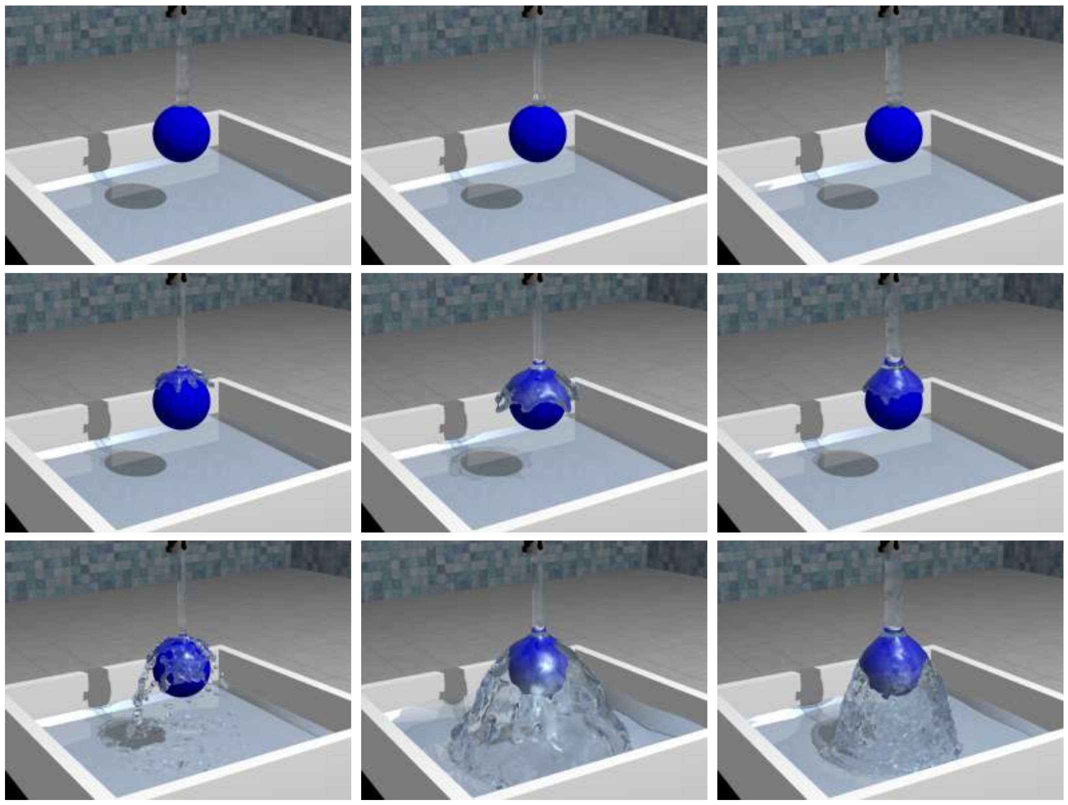


Figure 3.9: (Left Column) Our VOF method with a naive projection implementation which does not conserve volume. (Middle Column) Our VOF method with smear and pushout while replacing our velocity correction step with a standard Poisson solver. (Right Column) Our VOF method with proposed smear, pushout, and velocity correction steps. The middle and right columns conserve volume.

3.4 Partitioned Coupling

We utilize three different representations for water: besides the VOF representation on the KDSM, we also use both free particles and velocities on the background Eulerian Cartesian grid as is typical for the standard PLS method (see e.g. [36]). See Figure 3.8. Our partitioned coupling method consists of four major steps. In the first step, each of our three representations (VOF tetrahedra, particles, and Eulerian Cartesian grid) are advected forward in time. The method of Section 3.3.2 is used for the VOF tetrahedra, while the standard PLS method is used to advect the Eulerian grid velocities and to move the particles. In a second step, momentum is transferred between the three representations in order to maximize the visual efficacy of the results. Then, external forces are independently added to each representation, before projecting the velocity into a divergence free state acceptable to all three representations. The steps are summarized below:

1. Advection (each stage is independent)
   (a) VOF advection (Section 3.3.2)
   (b) Particle advection
   (c) Eulerian advection
2. Momentum transfer
   (a) Transfer momentum from VOF to Eulerian (optional)
   (b) Transfer momentum from Eulerian to VOF
   (c) VOF particle reincorporation
3. Add external forces (each stage is independent)
   (a) VOF external forces
   (b) Particle external forces
   (c) Eulerian external forces

4. Volume conservation
    (a) VOF volume conservation (Section 3.3.3)
    (b) Eulerian projection
        i. Eulerian particle reincorporation
        ii. Projection
    (c) Transfer momentum from Eulerian to VOF

Both the particles and the VOF tetrahedra carry accurate Lagrangian momentum information, as compared to the typically more smeared out velocities obtained using semi-Lagrangian advection (see e.g. [88]) on the Eulerian grid. Note that we allow VOF tetrahedra to overlap with the Eulerian water. Thus, we allow for the option to first transfer some momentum from the VOF tetrahedra to the Eulerian grid. Typically, this increases the turbulence near the boundaries of a moving creature. This is accomplished by iterating over tetrahedra with water and averaging their momentum with the values on the background Eulerian grid using an artist controllable multiplier. The result is used to overwrite the value on the Eulerian grid.

Next, the velocities of the Eulerian grid are used to overwrite the momentum value of any tetrahedron which has all four of its nodes inside the water surface representation of the Eulerian grid. The tetrahedron's volume is also set to be fully saturated with water. This overwrite operation does not use averaging since the background Eulerian grid has a full-fledged pressure solver that tracks velocities more accurately preserving various effects such as the circulation (discussed in Section 3.3.3). Importantly, cut cell tetrahedra are not overwritten allowing them to more accurately track volume and momentum close to the boundary of the creature. Higher ranks of tetrahedra could also be allowed to preserve their information if desired, although we did not experiment with this option.

Finally, any particle that lies within a VOF tetrahedron that contains water is deleted, and its volume and momentum are added to that tetrahedron. This allows particles to freely move through the region of space occupied by the KDSM only being reincorporated into the VOF representation when they impact water regions as defined by the VOF tetrahedra.

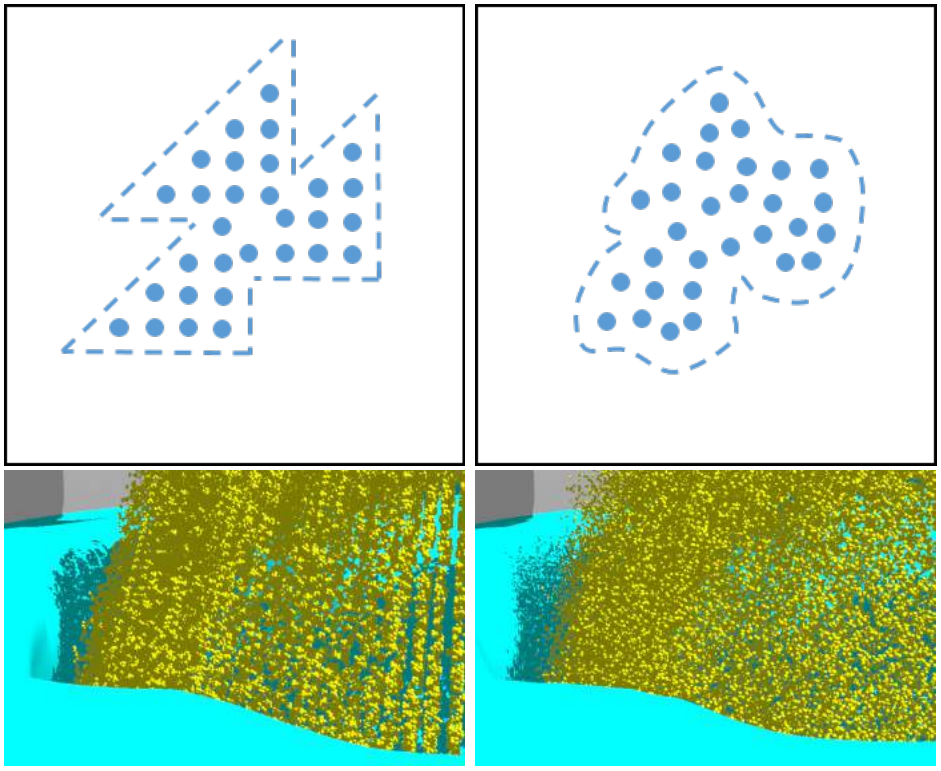


Figure 3.10: Top row compares the particle positions obtained with uniform versus jittered sampling emphasizing how well our eyes capture structured information (even when we do not want them to). Bottom row compares the two approaches for an actual simulation.

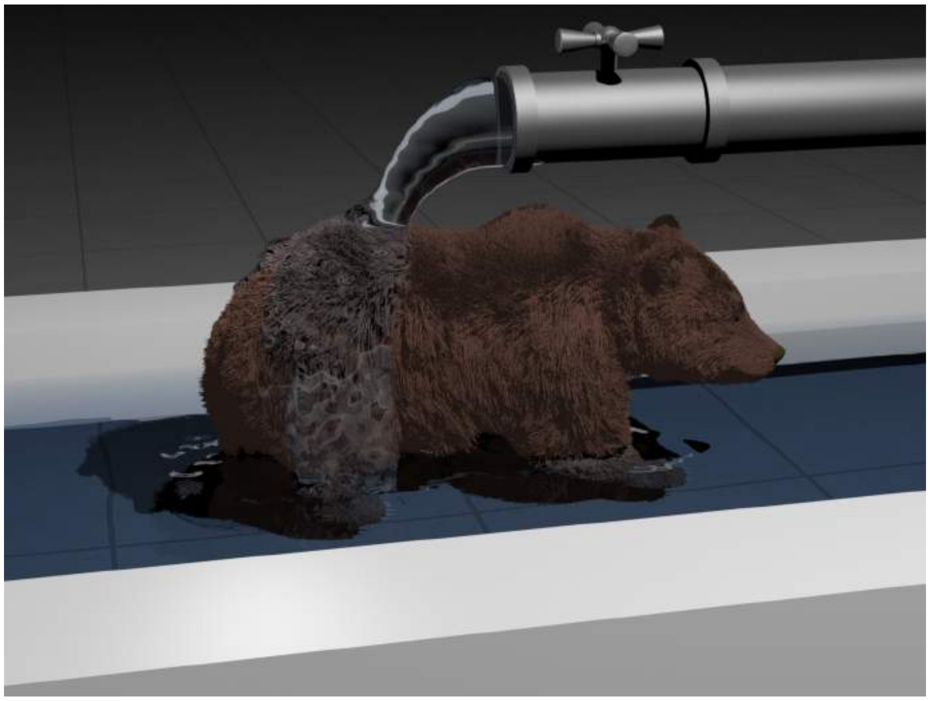


Figure 3.11: Our anisotropic porosity model is implemented to influence the VOF method on the KDSM accounting for both limited volume fraction and drag/adhesion yielding visually compelling results.

The volume conservation step starts out with the method proposed in Section 3.3.3, i.e. smear, pushout, and velocity correction, in order to create an adequate velocity for the VOF tetrahedra on the KDSM. Then, particles are reincorporated into the background Eulerian grid as Eulerian water when appropriate applying a local momentum force, altering the level set, and adding an expansion force similar to [62]. Following the standard PLS projection scheme, the results of the pressure solve are subsequently added to the Cartesian grid velocity in order to obtain a divergence free field. As a final step, the divergence free Eulerian grid velocities are used to overwrite the momentum in any tetrahedron that has all four of its nodes interior to the Eulerian grid water representation.

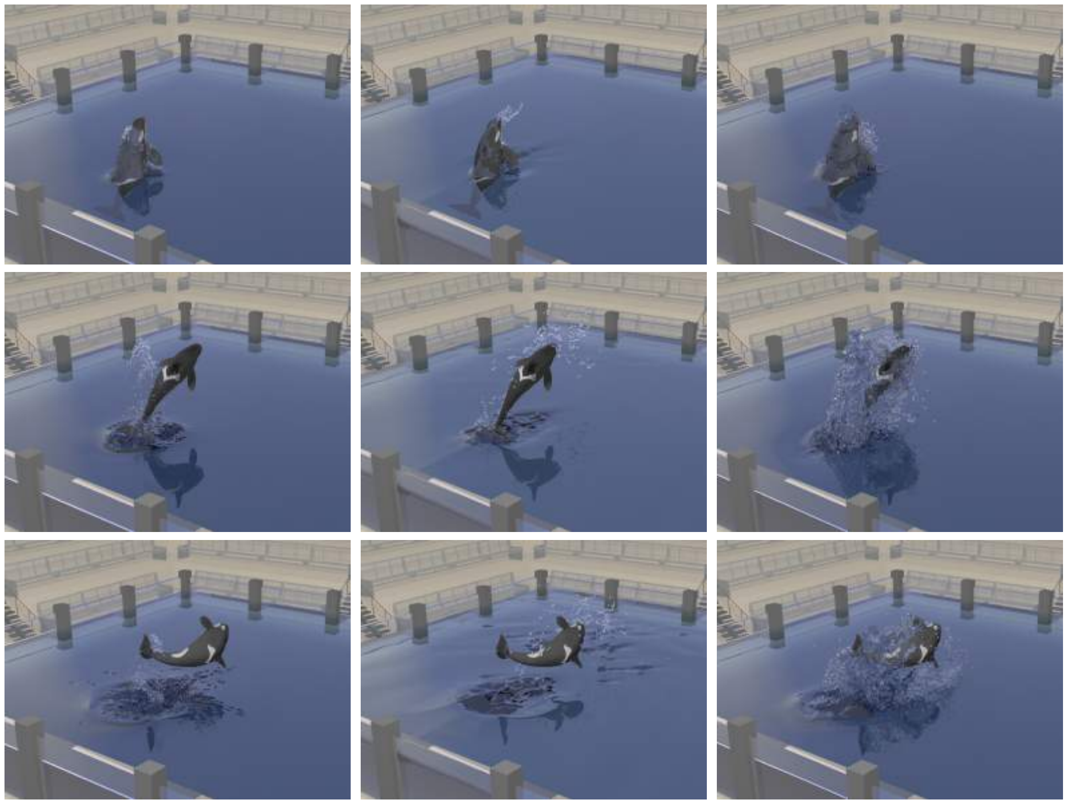


Figure 3.12: (Left) Whale breaching with the PLS method on high resolution grid. The whale pulls very little water along with it. (Middle) FLIP method, which also produces similar amount of sprays. (Right) Our method pulls more water into the air with the whale, producing interesting effects such as sheets and sprays.

3.4.1 Particle Generation

The automatic generation of particles in visually compelling locations by hybrid particle level set methods has been one of their strengths even predating the PLS method, see [42]. Thus, we devise a method similar in spirit for our ALE based VOF method on the KDSM. As discussed in Section 3.3.2, advection might dictate that water moves off of the KDSM. This occurs when part of a forward advected tetrahedron lies outside of the KDSM. This is detected by checking whether or not the point samples of the tetrahedron lie outside of the KDSM. Each point sample had already been assigned a certain amount of water to transport, so we use that water's volume and momentum to create a particle with appropriate radius and velocity. Note that we use a standard volume equation for a sphere, $V = 4/3\pi r^3$, to get radius. Since a straightforward approach leads to noticeable aliasing, we jitter the particle locations by a small amount–we used a fraction ranging from .1 to 1 multiplied by maximum edge length of a tetrahedron for our jitter magnitudes (see Figure 3.10). As discussed in Section 3.3.3, tetrahedra on the exterior boundary of the KDSM may contain excess water that needs to be transported off of the KDSM. In this scenario, there is no natural advection direction. Thus, we move the particles across the exterior face of the tetrahedron while also applying appropriate jittering. Note that when water leaves the KDSM, it always goes through particle phase before rejoining the level set.

3.4.2 Rendering

As is the case for many of the state-of-the-art Lagrangian methods, rendering smooth surfaces is quite difficult. Many authors have proposed various strategies, such as applying smoothing kernel on implicit surfaces as in [14], [110], [2], [87], [73], [99], [86], [74], [13], [69], explicitly tracking fluid surfaces as in [20], [19], and polygonalizing fluid surfaces as in [3], [5], [4], [102]. Since most of this research has been focused on rendering particles as opposed to triangles, we do not use a marching tetrahedra approach as in [32], [68] to render our VOF representation.

Instead, we propose a novel method of reconstructing water surface by running an advection-only PLS method while incorporating the VOF tetrahedra water and

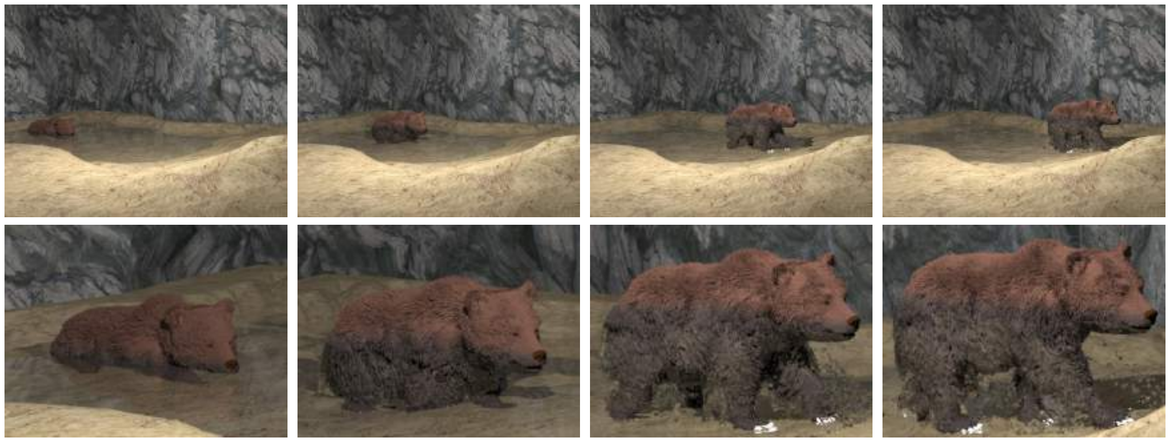


Figure 3.13: A bear walks out of a pool onto land, still carrying and dripping a large amount of water from its fur. Our anisotropic porosity model accounts for the correct volume fraction of water in the fur and uses adhesion to pull that water out of the pool with the bear, subsequently slowing dripping the water out of the fur. This example emphasizes the efficacy of the adaptivity of the KDSM as well as the ability to preserve volume and avoid disappearing water with our VOF method.

the PLS water from the simulation in order to achieve a smooth, temporally coherent surface with high level of detail. We allocate a refined Cartesian grid in order to reconstruct a new water surface, i.e. 2 to 4 times higher than simulation grid for our examples, and import the initial simulation data such as the PLS Cartesian grid and the KDSM. For each frame of our reconstruction scheme, we run a PLS advection, which advects the new water data, then transfer the next water information from the simulation to the refined grid. We also read the KDSM data and the VOF tetrahedra water data from the simulation, seeding negative particles to our recontruction grid along with VOF velocities. We average particle positions with a threshold similar to [104] and attract the particles that are near the level set of the simulation representing the water surface towards that level set in order to flatten out bumps created by the cut cell tetrahedra near the boundary of the level set. The number of seeded negative particles and their radii can be controlled via clamping with min/max value in order to influence the overall look of the resulting water surface. For instance, when a thick water surface is desired, one should increase the number or radii of particles that are seeded, while decreasing it will yield much more under-resolved level sets thereby the

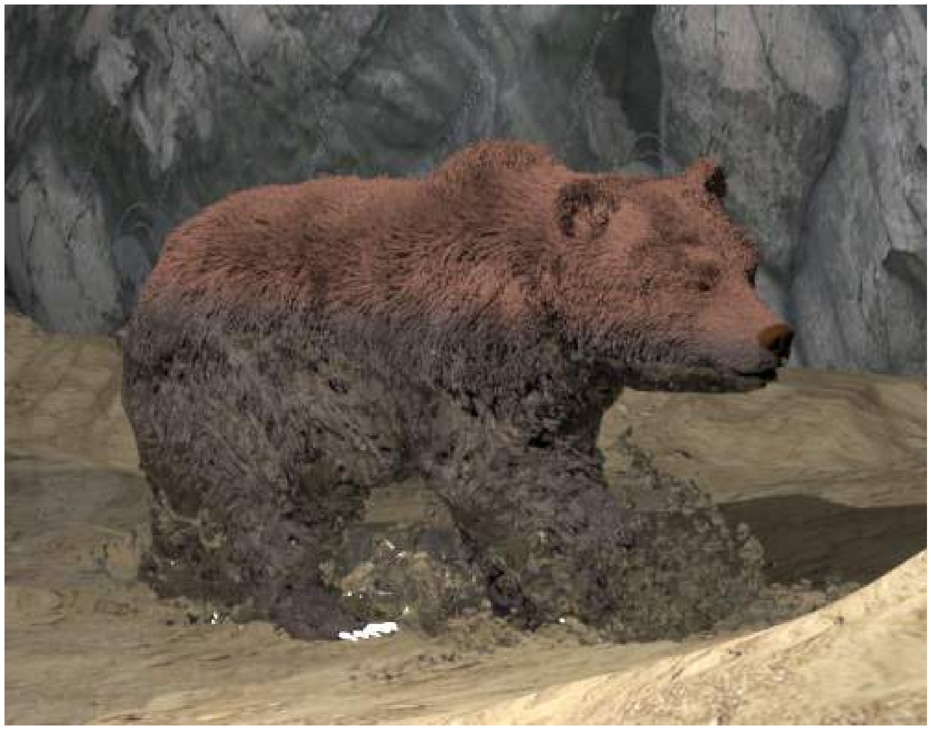


Figure 3.14: A close-up of the bear example, showing water sticking to fur and splashes generated from our VOF method.

less surface. Thus, one should increase the number of particles when the surfacing grid is highly refined otherwise the resulting surface will have many under-resolved regions. Then, we transfer the Cartesian grid data for water regions and solid boundaries, thereby potentially overwriting some regions where we modified with the KDSM data. Note that we add gravity only for the new water data, as simulation water data contains it already. We also utilized velocity extrapolation and setting boundary conditions from the PLS method to improve the results. Then, we reconstruct the surface and reseed to obtain a new level set surface and particles for our water. Finally, we run a diffusion in level set and smooth the normals for rendering, removing very high frequency jitters and undesirable bumps (see Figure 3.15). The result is a smooth and temporally coherent water surface, and we incorporate removed negative particle directly from the simulation if more sprays are needed. Our surface reconstruction performance significantly benefits from the absence of the pressure solver.

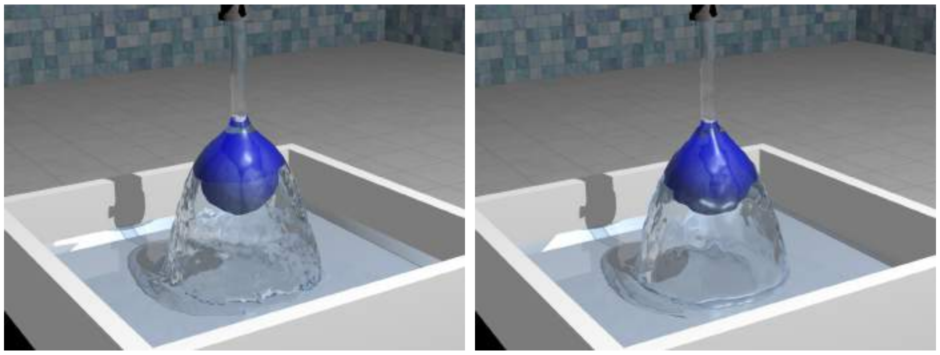


Figure 3.15: (Left) Anisotropic smoothing kernel in our ball example. (Right) Our surface reconstruction method.

3.5 Hair-Water Interaction

We embed hair particles in the KDSM and treat the hair using the KDSM as in [56] (we also refer the interested reader to [79], [59], [37], [38] for more discussion on hair-water interaction). Our hair-water approach is volumetric in nature, rasterizing multitude of hair representation into KDSM as opposed to [37], where they focus on a reduced model for individual hair strands. As a result, our method handles hair-water interaction with 540k hairs as opposed to 5k and 30k as given in [79] and [37], respectively. For each tetrahedron containing hair we precompute the volume fraction occupied by the hair and reduce the water that this tetrahedron may contain at saturation by this amount. This gives a very accurate representation of the porosity. We also compute the average direction of the hair strands in each tetrahedron, so that we may treat the porosity anisotropically. Essentially, more drag is applied orthogonal to the average direction of the hair strands. See Figures 3.11, 3.13, 3.14.

3.6 Results and Discussion

We ran our examples on a machine with a 3.06GHz CPU (12 cores) and 96GB RAM. KDSM generation for the whale and bear examples took 10 minutes per frame, and each frame is temporally independent so we ran them in parallel. The ball examples (Figures 3.3 and 3.7) took 1 minute and 2 seconds per frame to run with a 100x100x100

Eulerian grid, 5.6 million KDSM elements, and .9 million KDSM particles. The bear pour example (Figure 3.11) took 7 minutes and 3 seconds per frame with a 200x200x400 grid, 8.2 million KDSM elements, and 1.4 million KDSM particles. The bear walk example (Figure 3.13) took 4.5 minutes per frame with a 100x200x200 grid, 8.2 million KDSM elements, and 1.4 million KDSM particles. The whale example took 20 minutes per frame with a 200x300x200 grid, 8.5 million KDSM elements, and 1.4 million KDSM particles whereas the PLS method-only example took 29 minutes per frame using a 350x525x350 grid. We note the visual differences as a comparison in Figure 3.12 with FLIP method with a 200x300x200 grid, as well as Figure 3.17. If we run the PLS method for the whale example at an even higher resolution, we can eventually achieve higher quality results by carrying more water volume with the whale, but this would require significant time investment. We used Neumann boundary condition for solid boundaries for PLS method. For all our examples, we generated 5 to 35 samples per tetrahedron based on the quadrature formula.

There are fundamental limitations of ALE based methods especially regarding meshing problems, so we implement a couple of simple remedies below to fix the occasional degeneracy in order to run all of our examples robustly. Note that the animated creature can move in a way that inverts its elements or prevents volume preservation of the surrounding space unless the artist is very careful–most of issues appear near joints and are worsened by linear blend skinning. We only need to iterate a couple of times in the preprocessing stage to resolve most of these issues, and we disable any remaining degenerate elements (inverted or collapsed) so that they cannot participate in the VOF solver. Thus, whenever a sample point falls in degenerate elements, particles will be formed instead of the located element receiving water. While one could better prevent element inversion by using FEM or quasistatics, in practice our simple mass spring model was sufficient. Rarely when we cannot properly advect or enforce incompressibility during the simulation because a VOF tetrahedron is completely surrounded by the solid due to an extreme creature deformation, we simply keep the water in that tetrahedron in order to exactly conserve volume until the issue is resolved as the surrounding solid opens up.

Our method fully conserves volume in the KDSM, although floating point drift

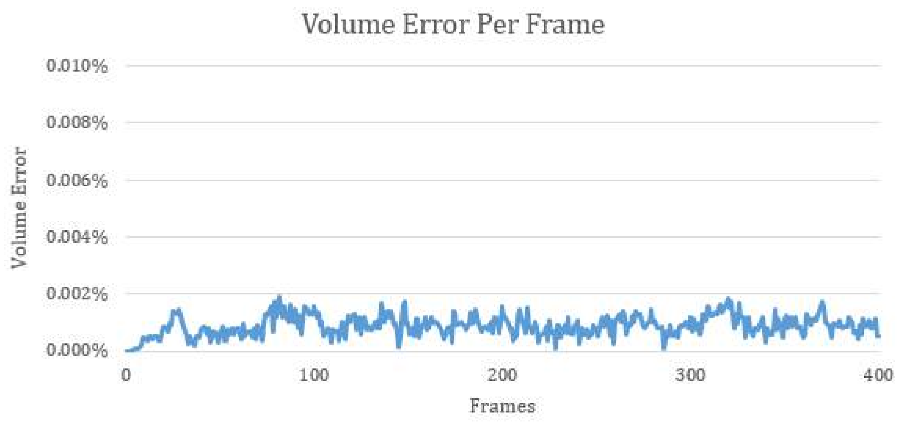


Figure 3.16: Volume error for example where a thin stream of water hits a ball (see top right of Figure 3.3).

causes small volume error throughout the simulation. We measured the volume error per frame for ball example in top right of Figure 3.3 for 400 frames. The average volume error per frame was 0.00089%, and the maximum error was 0.00189%.

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 Section 3.4 or by using a standard Poisson solver as discussed in Section 3.3.3.

As future work, one could implement a different solver such as [51], [50], or [26] to simulate fluid in the background grid or in the KDSM, and the adaptivity of our method will improve the accuracy of chosen method. In order to generalize our method to a pure FLIP/PIC/APIC variant, given that we already have a solver for the background Eulerian grid, the data transfer function would need to be rewritten in order to refer to the KDSM when the particle is inside of the KDSM, and the interpolation scheme would need to be modified to use barycentric weights for tetrahedron. Then, [8] could be used to handle non-advection steps. Thus, FLIP/PIC/APIC variant can benefit from the dense KDSM mesh instead of using the coarse background

Eulerian grid when the method transfers data from particles to the KDSM. We emphasize the technical insight that the coarse background grid captures a low frequency fluid surface whereas around the creature with high frequency boundaries we use the dense KDSM mesh to capture high frequency fluid motion. We chose the PLS method because it generates a very smooth surface, which is suitable for background motion, whereas our ALE based VOF method is more geared towards capturing detailed fluid motion by preserving volume to compensate for the PLS method's limitation. Additionally, one can subdivide on-the-fly if adaptive remeshing based on the fluid motion is desired.

3.7 Conclusions

We proposed a new fluid simulation framework for character-water and hair-water interaction using our novel volume conserving VOF method based on an adaptive tetrahedral mesh from the KDSM, which moves with the creature. We prebake the adaptivity of the ALE mesh, separating the nontrivial remeshing issue from the simulation phase and improving the robustness of our ALE based VOF method; we further preprocess auxiliary data wherever possible in order to make the simulation efficient and streamlined. A coarse background Eulerian grid and our fine ALE mesh are two way coupled using a partitioned approach which is fast, efficient, and straightforward to implement. We use our volume conserving VOF method only on the KDSM near

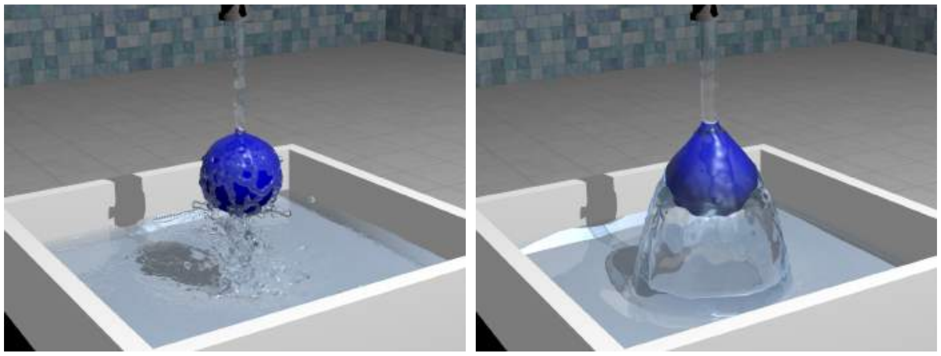


Figure 3.17: (Left) FLIP method on our ball example. (Right) Our method.

the creature while using a standard PLS method on the background Eulerian grid. We robustly implement interesting effects such as adhesion and anisotropic porosity. Moreover, we presented a novel water surfacing method to reconstruct the smooth, temporally coherent water surface. We demonstrated how the coarse background Eulerian grid captures the bulk behavior of the water, while our VOF method captures detailed water effects near the creature and the particles capture the spray—all of which make important contributions to the final result.

Chapter 4

Conclusions

This dissertation has presented KDSM as a data structure for hair and water simulation. The KDSM can deform along with the solid surface due to its Lagrangian nature while maintaining a consistent topology, as opposed to the grid which is fixed in space or particles which do not store topological information. It allows us to create hair-water framework which contributes in the following aspects:

Scalability: We improved the scalability of our framework by (1) precomputing numerous auxiliary information such as blendshape hair weights, adaptivity, topological information, adhesion coefficients, and porosity from a coarse KDSM, (2) using the layering framework, separating bulk motion and intricate motion of the simulation, and (3) providing a kinematic support for any simulated elements which are anchored to the KDSM to use weak springs.

Artist Control: We demonstrated KDSM based artist control such as (1) serving as a guide mesh for artists, (2) controlling the amount of KDSM deformation that can be applied to simulated elements, (3) containing air volume to let hair particles drift freely within the KDSM, and (4) producing adaptivity and detailed boundary conditions for the water simulation.

Flexibility: The KDSM is agnostic to the underlying simulation model, which allows us to create the flexible framework for both hair and water simulation on top of the KDSM.

Here are contributions from our hair-water simulation system for hair animation, robust volume-conserving water simulation, and hair-water interaction based on the KDSM. First, we demonstrated our novel hair pipeline and hair-water interaction scheme with the PLS method. Next, we proposed a robust water simulation technique that preserves volume for character-water interaction, which produces improved hair-water interaction. Finally, we showed how our framework captures high-quality hair, water, and hair-water effects while scaling well to millions of hairs and very large fluid domains, offering powerful artist control, and providing a flexible framework integrating multiple methods. Notably, the KDSM is intrinsically part of the animation and does not dictate any particular approach to simulation; this feature allows developers/artists to use their favorite techniques in place of our mass spring model, the PLS method, and even our VOF method. We mainly benefited from the technical insight that we can use different models for different scales. For instance, the finite element method or position based dynamics can be used instead of mass spring as a constitutive model for kinematic/dynamic skinning, a reduced hair model can be used instead of mass spring hair in place of our individual hair model, and PIC/FLIP/APIC can be used instead of the PLS or our VOF method.

We present below areas for extension of this work.

4.1 Rendering

The rendering for hair-water effects proved to be a non-trivial task, due to the sheer number of elements to render and the various quality issues. We noticed various post-processing/rendering artifacts (e.g. noise, temporal incoherency, and bumps), which degrades the overall quality of the final video. For our hair-water interaction examples, water naturally generates splashes which hides some artifacts with the help of reflection/refraction, but these were often visible in our debug rendering mode. Our water reconstruction scheme improves upon the state-of-the-art, but this area still remains to be explored and improved upon, possibly by extending our reconstruction scheme.

4.2 Hair Framework Improvement

The dynamic skinning and blendshape hair can be extended to achieve hair effects with higher fidelity. For instance, swaying the bear back and forth underwater can be improved by running more than one copy of the simulated dynamic tetrahedral mesh. Because of running multiple versions of dynamic skinning with the blendshape hair, our framework would be able to support more than one type of dynamics for the bulk and intricate hair motion. Finally, multiple versions of hairs can be blended as a post-process to achieve various motions of hair.

4.3 Cloth Simulation

Since KDSM preserves inertial effects around the boundaries of an object, it will benefit cloth simulation. Kinematic skinning can be used to deform cloth based on the deformation of the character. Dynamic skinning can be used to handle external forces such as wind or water drag on cloth. A blendshape variant can be applied to cloth mesh to create or add any wrinkles or deformations based on the change in character's pose. When cloth undergoes collisions, a subset of cloth mesh can be selected and any constitutive model can be applied to the selected triangles while the remaining cloth mesh is embedded to the KDSM to retain its intended motion. In conclusion, cloth simulation seems to be a natural direction in which to extend our framework.

4.4 Smoke Simulation

We have demonstrated water simulation using the KDSM, so smoke simulation will be a straightforward extension. [39], [40], [55] cover many relevant ALE smoke implementation details, which are relevant here. Most of the second chapter of this dissertation can be used except the projection stage because smoke is a compressible fluid phenomenon unlike water which is incompressible.

4.5 Tree Simulation

Since trees have numerous leaves whose movements are mainly governed by the motion of branches, our KDSM can be very useful in running tree simulation with millions of leaves. The low frequency motion of leaves caused by branch movements can be implemented as in bulk motion, and high frequency leaf motion can be represented by either precomputed blendshape or simulated individually with a reduced model. Furthermore, additional wind or water simulation can be coupled with tree simulation to produce a forest scene with numerous trees interacting with strong wind gusts from a storm, smoke from wildfire, or waves caused by a tsunami.

4.6 Artist Control for Water Simulation

Currently, our method only supports adhesion force based controls for water simulation with our novel VOF method. Our method provides enough control for water to robustly implement hair-water interaction via our anisotropic porosity for hairs, but an arbitrary control without porosity-like quantities could be nontrivial. Integrating [64] would be a good direction to work in to improve controllability of both our VOF and PLS solvers per keyframe.

4.7 Real-Time Hair/Water for Games

The kinematic skinning and blendshape hair can be implemented in GPU to achieve hair effects in real-time for games. Our VOF solver can be weakly enforced (volume conservation enforced with a few specified bandwidth) in order to improve performance. Since character animation for games are precomputed and simply interpolated in real-time, the KDSM can be processed in the same manner and be used to compute hairs and improve performance of water simulation. NVIDIA HairWorks implemented an efficient hair/fur model for game characters/creatures such as humans, wolves, tigers, and monsters, and our pipeline can further improve NVIDIA's pipeline to increase the realism of AAA games such as Witcher, Far Cry, Call of Duty, and Tomb Raider series.

4.8 Adaptive Mesh for Neural Network

Last, our framework can be useful as a spatially adaptive mesh for machine learning/deep learning techniques, especially serving as a grid for convolutional neural network. The KDSM contains inertial effects of a character, and pose based machine learning techniques can be integrated with our method to learn various effects around the boundaries based on the pose, and the learned model could be applied to the tetrahedra of the KDSM mesh (most likely after a few subdivisions) for an intended effect. The KDSM can be useful in learning deformations of hair, cloth or skin based on a character's poses because of the consistent topology aspect of the KDSM.

Bibliography

[1] M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM TOG*, 36, 2017.

[2] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 26(3), 2007.

[3] G. Akinci, N. Akinci, M. Ihmsen, and M. Teschner. An efficient surface reconstruction pipeline for particle-based fluids. In *Proceedings of Virtual Reality Interactions and Physical Simulations*, 2012.

[4] G. Akinci, N. Akinci, E. Oswald, and M. Teschner. Adaptive surface reconstruction for sph using 3-level uniform grids. *WSCG 2013*, 2013.

[5] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum*, 2012.

[6] Nadir Akinci, Gizem Akinci, and Matthias Teschner. Versatile surface tension and adhesion for sph fluids. *ACM Trans. Graph.*, 32(6):182:1–182:8, November 2013.

[7] Dicko Ali-Hamadi, Tiantian Liu, Benjamin Gilles, Ladislav Kavan, François Faure, Olivier Palombi, and Marie-Paule Cani. Anatomy transfer. In *ACM SIGGRAPH Asia 2013 papers*, SIGGRAPH ASIA '13, pages 188:1–188:8, 2013.

[8] R. Ando, N. Thürey, and C. Wojtan. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. (Proc. SIGGRAPH 2013)*, July 2013.

[9] Kenichi Anjyo, Yoshiaki Usami, and Tsuneya Kurihara. A simple method for extracting the natural beauty of hair. In *Comput. Graph. (Proc. ACM SIGGRAPH 92)*, volume 26, pages 111–120. ACM, 1992.

[10] Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. Animating hair with loosely connected particles. In *Comput. Graph. Forum*, volume 22, pages 411–418. Wiley Online Library, 2003.

[11] C. Batty, S. Xenos, and B. Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. In *Comput. Graph. Forum (Eurographics Proc.)*, volume 29, pages 695–704, 2010.

[12] Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. Graph.*, 30(1):6, 2011.

[13] H. Bhatacharya, Y. Gao, and A. Bargteil. A level-set method for skinning animated particle data. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, 2011.

[14] J. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph. (Proc. SIGGRAPH 1982)*, 1982.

[15] Ugo Bonanni, Melanie Montagnol, and Nadia Magnenat-Thalmann. Multilayered visuo-haptic hair simulation. *The Vis. Comput.*, 24(10):901–910, 2008.

[16] Stephen D Bowline and Zoran Kacic-Alesic. A unified dynamics pipeline for hair, cloth, and flesh in Rango. In *ACM SIGGRAPH 2011 Talks*. ACM, 2011.

[17] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modelling surface tension. *J. Comput. Phys.*, 100:335–353, 1992.

[18] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 28–36, 2003.

[19] T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 47:1–47:9, 2010.

[20] T. Brochu and R. Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493, 2009.

[21] Armin Bruderlin. A method to generate wet and broken-up animal fur. In *Comput. Graph. and App., 1999. Proc. Seventh Pacific Conf. on*, pages 242–249. IEEE, 1999.

[22] M. Chai, C. Zheng, and K. Zhou. Adaptive skinning for interactive hair-solid simulation. *TVCG*, 2016.

[23] Menglei Chai, Changxi Zheng, and Kun Zhou. A reduced model for interactive hairs. *ACM Trans. Graph.*, 33(4):124, 2014.

[24] J. T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *Proc. ACM SIGGRAPH Symp. on Comput. Anim.*, pages 77–80, 2002.

[25] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 219–228, 2007.

[26] Nuttapong Chentanez, Matthias Müller, and Tae-Yong Kim. Coupling 3d eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena. pages 1–10, 2014.

[27] B. Choe, M.G. Choi, and H-S. Ko. Simulating complex hair with robust collision handling. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 153–160, 2005.

[28] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 2(1):12–26, 1967.

[29] A.J. Chorin. Numerical solution of the Navier-Stokes Equations. *Math. Comput.*, 22(1):745–762, 1968.

[30] M. Cong, M. Bao, J. L. E, K. S. Bhat, and R. Fedkiw. Fully automatic generation of anatomical face simulation models. In *Proc. of the 14th ACM SIGGRAPH / Eurographics Symp. on Comput. Anim.*, pages 175–183, 2015.

[31] Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *ACM Trans. Graph.*, volume 30, page 139. ACM, 2011.

[32] A. Doi and A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Trans. Inf. & Syst.*, E74-D:214–224, 1991.

[33] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. An adaptive discretization of incompressible flow using a multitude of moving Cartesian grids. *J. Comput. Phys.*, 254(0):107 – 154, 2013.

[34] R.E. English, L. Qiu, Y. Yu, and R. Fedkiw. Chimera grids for water simulation. In *SCA '13: Proceedings of the 2013 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2013.

[35] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183:83–116, 2002.

[36] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[37] Y. Fei, H. T. Maia, C. Batty, C. Zheng, and E. Grinspun. A multi-scale model for simulating liquid-hair interactions. *ACM Trans. Graph.*, 36(4), 2017.

[38] Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.*, 37(4):51:1–51:16, August 2018.

[39] B. Feldman, J. O'Brien, and B. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):904–909, 2005.

[40] B. Feldman, J. O'Brien, B. Klingner, and T. Goktekin. Fluids in deforming meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 255–259, 2005.

[41] Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. Narrow band flip for liquid simulations. *Comput. Graph. Forum*, 35(2):225–232, May 2016.

[42] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001*, pages 23–30, 2001.

[43] N. Foster and D. Metaxas. Realistic animation of liquids. *Graph. Models and Image Processing*, 58:471–483, 1996.

[44] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM TOG*, 22(3):871–878, 2003.

[45] Rajeev Gupta and Nadia Magnenat-Thalmann. Interactive rendering of optical effects in wet hair. In *Proc. of the 2007 ACM Symp. on Virt. Reality Software and Tech.*, pages 133–140. ACM, 2007.

[46] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Comput. Graph. Forum*, 20(3), 2001.

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

[48] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. Single-view hair modeling using a hairstyle database. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2015)*, 34(4), July 2015.

[49] Hayley Iben, Mark Meyer, Lena Petrovic, Olivier Soares, John Anderson, and Andrew Witkin. Artistic simulation of curly hair. In *Proc. of the 12th ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 63–71. ACM, 2013.

[50] Markus Ihmsen, Nadir Akinci, Gizem Akinci, and Matthias Teschner. Unified spray, foam and air bubbles for particle-based fluids. *Vis. Comput.*, 28(6-8):669–677, 2012.

[51] C. Jiang, C. Schroeder, and J. Teran. An affine particle-in-cell method. *ACM Trans. Graph. (Proc. SIGGRAPH 2015)*, 2015.

[52] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In *Comput. Graph. (Proc. ACM SIGGRAPH 90)*, pages 271–280. ACM, 1989.

[53] Danny M Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Trans. Graph.*, 33(4):123, 2014.

[54] Theodore Kim, Nils Thürey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–6, 2008.

[55] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 25(3):820–825, 2006.

[56] M. Lee, D. Hyde, M. Bao, and R. Fedkiw. A skinned tetrahedral mesh for hair animation and hair-water interaction. *IEEE TVCG*, 2018.

[57] M. Lentine, M. Aanjaneya, and R. Fedkiw. Mass and momentum conservation for fluid simulation. In *SCA '11: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 91–100, 2011.

[58] M. Lentine, M. Cong, S. Patkar, and R. Fedkiw. Simulating free surface flow with very large time steps. In *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim. 2012*, pages 107–116, 2012.

[59] W. Lin. Coupling hair with smoothed particle hydrodynamics fluids. *Proc. of VRIPHYS*, 2014.

[60] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35:995–1010, 2006.

[61] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23:457–462, 2004.

[62] F. Losasso, J. O. Talton, K. Nipun, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE TVCG*, 14:797–804, July 2008.

[63] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. Detail preserving continuum simulation of straight hair. In *Proc. SIGGRAPH 2009*, pages 385–392, 2009.

[64] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23(3):449–456, 2004.

[65] V. Mihalef, D. Metaxas, and M. Sussman. Animation and control of breaking waves. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 315–324, 2004.

[66] V. Mihalef, B. Unlusu, D. Metaxas, M. Sussman, and M. Hussaini. Physics based boiling simulation. In *SCA '06: Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 317–324, 2006.

[67] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Mesh. Roundtable*, pages 103–114, 2003.

[68] H. Müller and M. Wehle. Visualization of implicit surfaces using adaptive tetrahedrizations. 1997.

[69] M. Müller and N. Chentanez. Solid simulation with oriented particles. *ACM TOG*, 30(4):92:1–92:10, 2011.

[70] Matthias Müller and Nuttapong Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 85–92. Eurographics Association, 2010.

[71] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Air meshes for robust collision handling. *ACM Trans. Graph.*, 34(4):133:1–133:9, July 2015.

[72] Matthias Müller, Tae-Yong Kim, and Nuttapong Chentanez. Fast simulation of inextensible hair and fur. *VRIPHYS*, 12:39–44, 2012.

[73] K. Museth, M. Clive, and N. B. Zafar. Blobtacular: Surfacing particle systems in "pirates of the caribbean 3". *SIGGRAPH Sketches*, 2007.

[74] J. Onderik, M. Chladek, and R. Durikovic. Sph with small scale details and improved surface reconstruction. *SCCG*, 2011.

[75] Lena Petrovic, Mark Henne, and John Anderson. Volumetric methods for simulation and rendering of hair. *Pixar Anim. Studios*, 2(4), 2005.

[76] Eric Plante, Marie-Paule Cani, and Pierre Poulin. Capturing the complexity of hair motion. *Graph. Models*, 64(1):40–58, 2002.

[77] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *J. Comput. Phys.*, 141:112–152, 1998.

[78] Robert E Rosenblum, Wayne E Carlson, and Edwin Tripp. Simulating the structure and dynamics of human hair: modelling, rendering and animation. *J. Vis. and Comput. Anim.*, 2(4):141–148, 1991.

[79] W. Rungjiratananon, Y. Kanamori, and T. Nishita. Wetting effects in hair simulation. In *Comput. Graph. Forum*, volume 31, pages 1993–2002. Wiley Online Library, 2012.

[80] R Sánchez-Banderas, H Barreiro, I García-Fernández, and M Pérez. Real-time inextensible hair with volume and shape. In *Congreso Español de Informática Gráfica, CEIG'15*, 2015.

[81] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 27(3):64.1–64.11, August 2008.

[82] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 81–90, 2007.

[83] Eftychios Sifakis, Sebastian Marino, and Joseph Teran. Globally coupled collision handling using volume preserving impulses. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 147–153. Eurographics Association, 2008.

[84] Maryann Simmons, Kelly Ward, Hidetaka Yosumi, Hubert Leo, and Xinmin Zhao. Directing hair motion on tangled. In *ACM SIGGRAPH 2011 Talks*, page 41. ACM, 2011.

[85] Maryann Simmons and Brian Whited. Disney's hair pipeline: crafting hair styles from design to motion. In *Eurographics 2014 Industrial Presentations*, 2014.

[86] F. Sin, A. W. Bargteil, and J. K. Hodgins. A point-based method for animating incompressible flow. In *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 247–255, 2009.

[87] B. Solenthaler, J. Schlüfli, and R. Pajarola. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds*, 2007.

[88] J. Stam. Stable fluids. In *Proc. of SIGGRAPH 99*, pages 121–128, 1999.

[89] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comput. Phys.*, 148:81–124, 1999.

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

[91] Pascal Volino and Nadia Magnenat-Thalmann. Animating complex hairstyles in real-time. In *Proc. of the ACM Symp. on Virt. Reality Software and Tech.*, pages 41–48. ACM, 2004.

[92] Pascal Volino and Nadia Magnenat-Thalmann. Real-time animation of complex hairstyles. *IEEE Trans. on Vis. and Comput. Graph.*, 12(2):131–142, 2006.

[93] Kelly Ward, Nico Galoppo, and Ming Lin. Interactive virtual hair salon. *Presence: Teleoperators and Virt. Env.*, 16(3):237–251, 2007.

[94] Kelly Ward, Nico Galoppo, and Ming C. Lin. Modeling hair influenced by water and styling products. In *Proc. of Comput. Anim. and Social Agents (CASA)*, pages 207–214, 2004.

[95] Kelly Ward, Nico Galoppo, and Ming C. Lin. Simulating and rendering wet hair. In *SIGGRAPH 2004 Sketches*, page 42. ACM Press, 2004.

[96] Kelly Ward, Ming C. Lin, Joohi Lee, Susan Fisher, and Dean Macri. Modeling hair using level-of-detail representations. In *Proc. of Comput. Anim. and Social Agents (CASA)*, page 41, 2003.

[97] Kelly Ward, Maryann Simmons, Andy Milne, Hidetaka Yosumi, Xinmin Zhao, and Walt Disney Animation Studios. Simulating rapunzel's hair in disney's tangled. In *SIGGRAPH Talks*, 2010.

[98] R. Weinstein, E. Guendelman, and R. Fedkiw. Impulse-based control of joints and muscles. *IEEE Trans. on Vis. and Comput. Graph.*, 14(1):37–46, 2008.

[99] B. W. Williams. *Fluid Surface Reconstruction from Particles.* PhD thesis, The University of British Columbia, 2008.

[100] Keith Wilson, Aleka McAdams, Hubert Leo, Maryann Simmons, and Walt Disney Animation Studios. Simulating wind effects on cloth and hair in Disney's Frozen. In *SIGGRAPH Talks*, pages 48–1, 2014.

[101] Kui Wu and Cem Yuksel. Real-time hair mesh simulation. In *Proc. of the 20th ACM SIGGRAPH Symp. on Interactive 3D Graph. and Games*, pages 59–64. ACM, 2016.

[102] W. Wu, H. Li, T. Su, H. Liu, and Z. Lv. Gpu-accelerated sph fluids surface reconstruction using two-level spatial uniform grids. 33, 07 2016.

[103] Hongyi Xu and Jernej Barbič. Pose-space subspace dynamics. *ACM Trans. Graph.*, 35(4):35, 2016.

[104] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pages 217–225, 2010.

[105] O. Zarifi and C. Batty. A positive-definite cut-cell method for strong two-way coupling between fluids and deformable bodies. In *Proceedings of the 2016 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[106] Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. An advection-reflection solver for detail-preserving fluid simulation. *ACM Trans. Graph.*, 37(4):85:1–85:8, July 2018.

[107] L. Zhang, T. Cui, and H. Liu. A set of symmetric quadrature rules on triangle and tetrahedra. 2009.

[108] X. Zhang, M. Li, and R. Bridson. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Trans. Graph.*, 35(4), 2016.

[109] B. Zhu, E. Quigley, M. Cong, J. Solomon, and R. Fedkiw. Codimensional surface tension flow on simplicial complexes. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 33(4):111:1–111:11, 2014.

[110] Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):965–972, 2005.