A Skinned Tetrahedral Mesh for Hair Animation and Hair-Water Interaction

Minjae Lee, David Hyde, Michael Bao, Ronald Fedkiw

Abstract—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.

Index Terms—Computer Graphics, Hair, Skinning, Tetrahedral Mesh, Blendshape, Water

1 Introduction

The seminal rendering of the teddy bear in [1] immediately piqued a great deal of interest in hair animation and simulation, see [2], [3]. 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 [4], Rapunzel in *Tangled* [5], [6], Merida in *Brave* [7], Elsa in *Frozen* [8], [9], 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 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 control by the animator. This is especially useful in handling intricate hair motion such

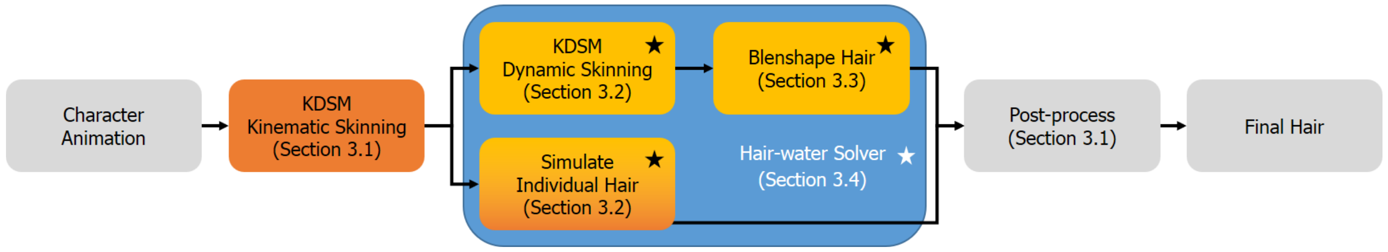

Character Animation
KDSM Kinematic Skinning (Section 3.1)
KDSM Dynamic Skinning (Section 3.2) ★
Simulate Individual Hair (Section 3.2) ★
Blenshape Hair (Section 3.3) ★
Hair-water Solver (Section 3.4) ★
Post-process (Section 3.1)
Final Hair

Fig. 1. 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.

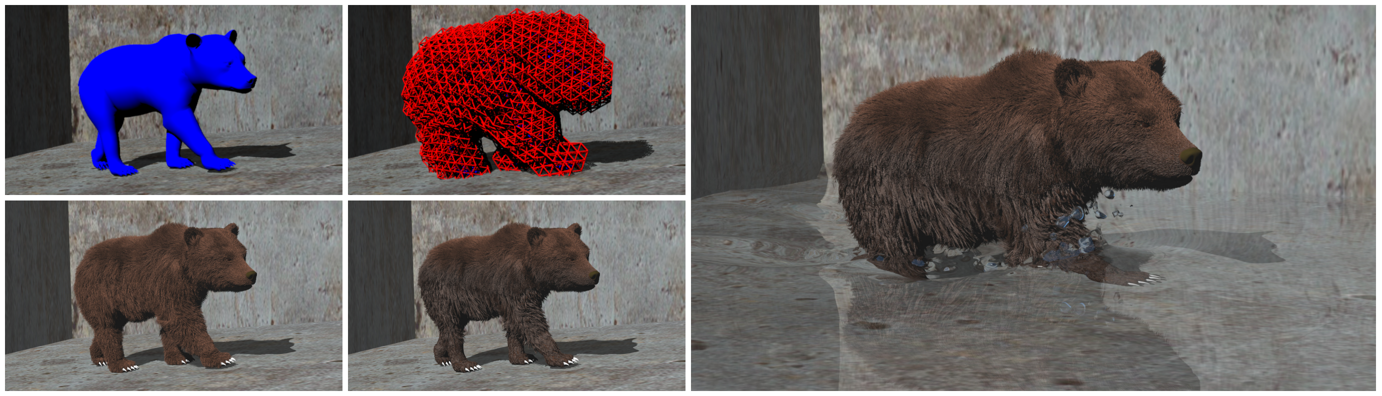


Fig. 2. Given a skinned animation (top left), we skin a tetrahedral mesh to follow that animation while including the effects of inertia, deformation, pinching, etc. (top right). Subsequently, hair embedded in the tetrahedral mesh moves and deforms to follow that mesh with the potential for additional 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).

as wetting and clumping of hairs. Finally, noting that [10] 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 hardbound 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 1.

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. [11], [12], [4]. Methods that use guide hairs, see e.g. [13], [14], [15], [16], 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 [17] to obtain impressive results. [18] 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. [19], [20], [21], [7], [22], in which case it becomes important to consider physical interactions between actual hairs (see e.g. [23]) as opposed to interactions based on volumetric continuum assumptions for guide hairs.

Motivated by [24], [25], [26], 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 [27], [28], where the hair was embedded in a simulated volumetric lattice enclosing the head. [29] pointed out that hair simulated in this fashion too closely follows the continuum and subsequently proposed a two-layered approach similar to [30] and [31]. We instead address this by utilizing a hybrid solids simulation framework [32] with soft bindings that allow our hairs to drift away from their

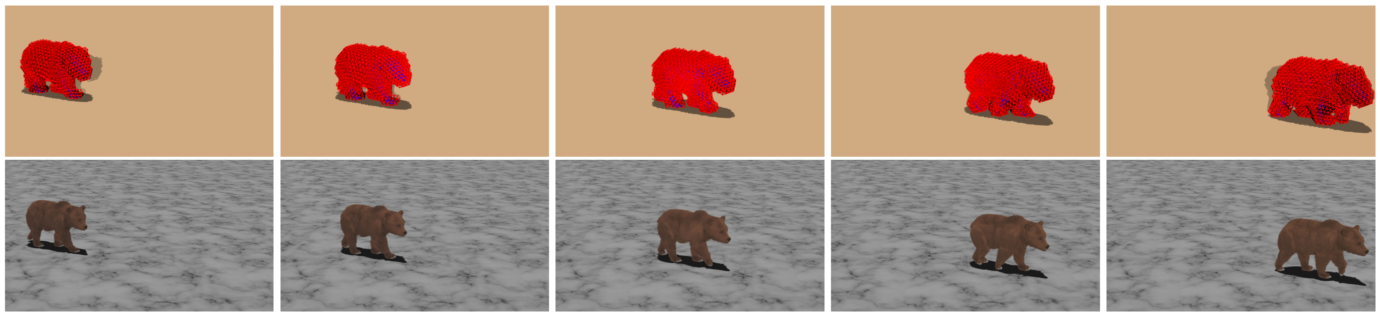


Fig. 3. (Top) We create a kinematically deforming skinned mesh (KDSM) that follows the animation and includes the effects of inertia, deformation, swinging motions, etc. (Bottom) Subsequently, any reference frame hairstyle can be skinned throughout the animation by simply following its embedding in the KDSM via barycentric coordinates.

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

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 3.1. The second layer involves either dynamic hair simulation (Section 3.2) or blendshape hairs (Section 3.3), coupled with a hair-water solver as discussed in Section 3.4. Finally, length preservation and pushout are applied as a post-process per frame as discussed in Section 3.1.

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 [33], [34] along with the feature detection from [34] if the mesh topology is not consistent from frame to frame. One could perhaps alternatively use an approach like that of [35] 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 [36] 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 [36] 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 3 (top), Figure 5 (top left), and Figure 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 [27], [28] except that the base of our mesh is driven by an underlying morph of

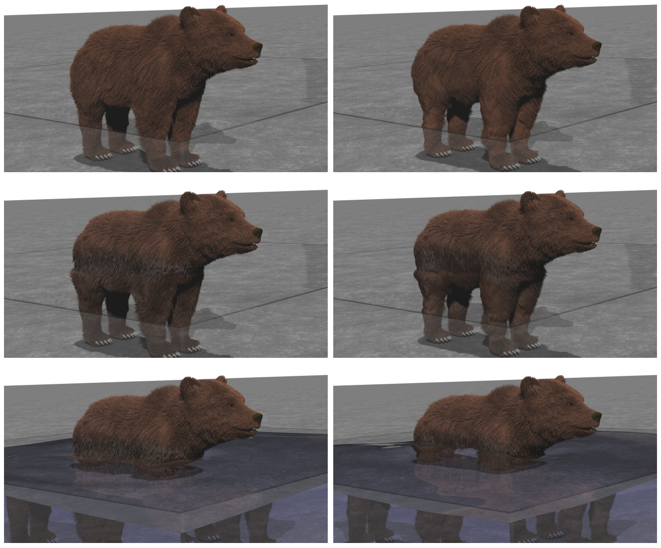


Fig. 4. Treating hair volumetrically, a simulated dynamic tetrahedral mesh is attached to the KDSM for structure and support, and sways back and forth underwater using the water velocity to apply drag. (Top) No blendshape hair. (Middle) Blendshape hair. (Bottom) Blendshape hair with water.

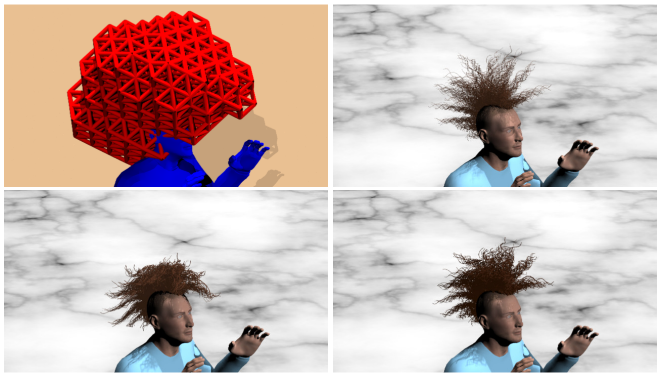


Fig. 5. (Top left) KDSM. (Top right) No simulation. (Bottom left) Simulation with weak zero-length spring attachments to the KDSM. (Bottom right) Simulation with stronger zero-length spring attachments to the KDSM.

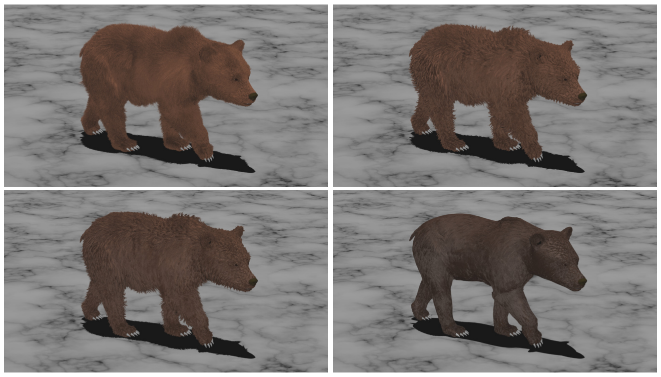


Fig. 6. Starting from the rest hairstyle (top left), individual hair and volumetric hair simulations as well as procedural techniques were used to create clumped (top right), sagged (bottom left), and matted (bottom right) hairstyles, which were then all bound to the animation sequence from Figure 3 using that KDSM.

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 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

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

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 [32]. 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 [37], [38]). Interpenetrations with the character are handled using the pushout method from [39] in order to preserve the style of the hair. All 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 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.

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 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.

3.2.1 *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

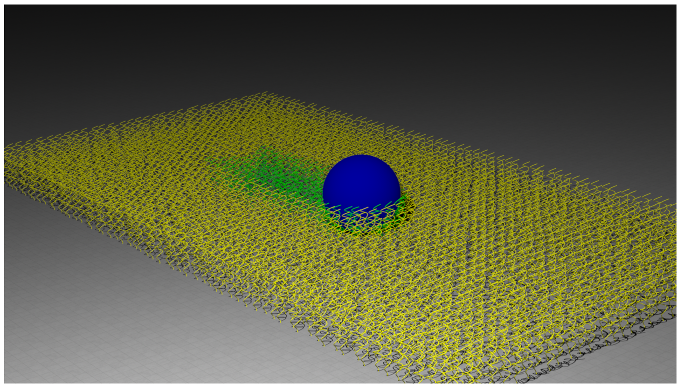


Fig. 7. Our framework supports selective activation of individual hairs for simulation in order to only activate hairs that are undergoing collision. Here, fully kinematic hairs are shown in yellow while hairs activated for individual simulation are shown in green.

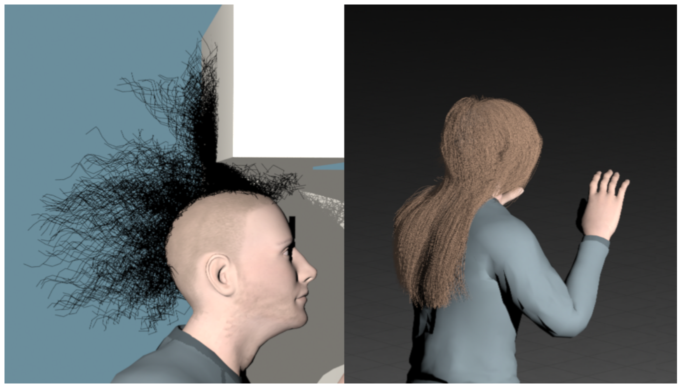


Fig. 8. (Left) Hairs are individually simulated for high fidelity collision handling. (Right) Individual hair simulation is also used for long hairstyles. Both examples benefit from structure and support provided by the KDSM.

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 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.

3.2.2 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 [32] connecting a zero-length spring between

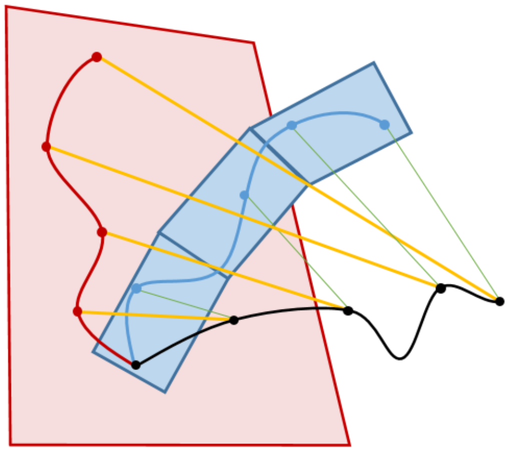


Fig. 9. Visualization of structure contributing to the final hair shape: the simulated hair (black curve) is connected to embedded positions within the KDSM (red box) and to shape-preserving tetrahedra (blue boxes) by zero-length springs (yellow and green, respectively).

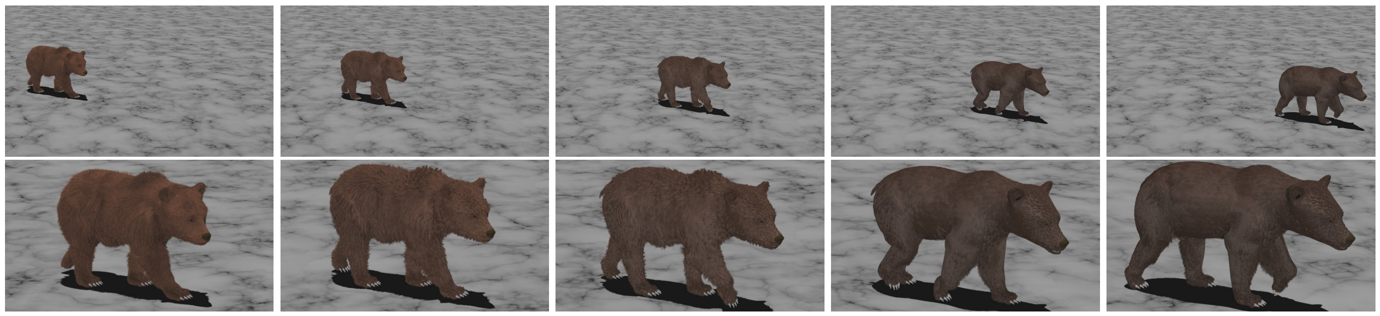
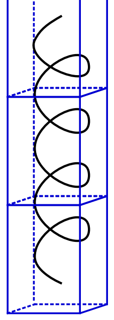


Fig. 10. (Top row) Using the hairstyles from Figure 6 and the KDSM from Figure 3 along with our blendshape system, we make a creature grow progressively wetter as it walks. (Bottom row) Close ups of top row.

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 [19]. Although we use the mass-spring method of [19] for hair-hair interactions and collisions for the examples shown in Figure 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 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 8 depicts a long hairstyle from [40] 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 [19], 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 [41], 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 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 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 [18], as in Figure 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

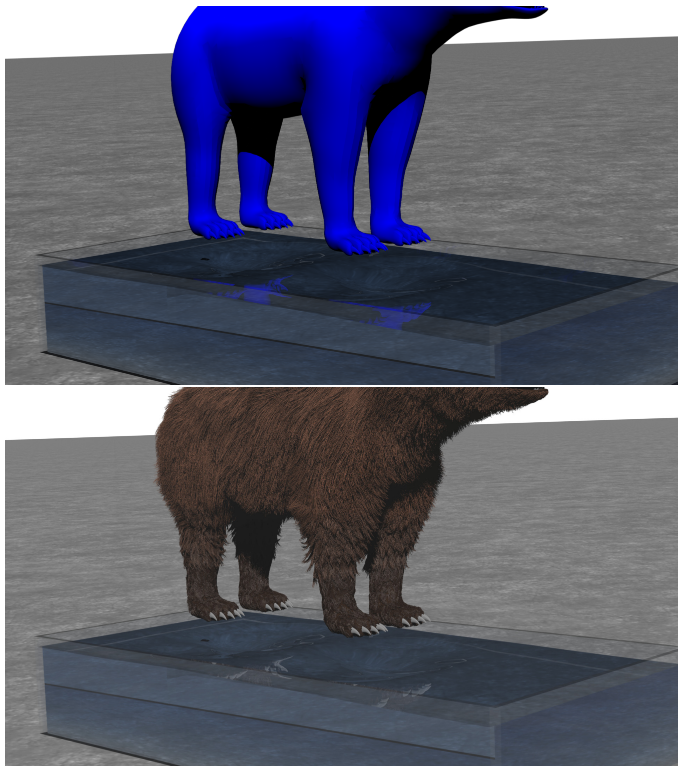


Fig. 11. (Top) First a water simulation is performed, ignoring the hair. (Bottom) Then in a post-process, a procedurally generated wet map is used in the blendshape system to locally (in space and time) interpolate between various hairstyles such as those in Figure 6.

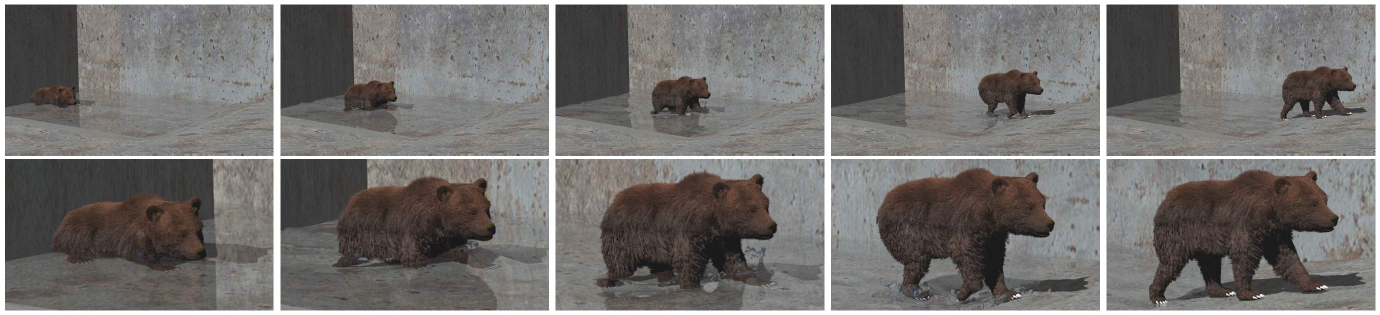


Fig. 12. (Top) Creature walking out of water. (Bottom) Close ups of top row.

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 [42]) 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 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 [27], [28], 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. [43], [14], [44], [45], [19]) 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 6 top left) or the new clumped hairstyle (see Figure 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 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 6 bottom right).

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 6 for dry, clumped, sagged, and 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 [43], [44], [45], [46], [47], 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 6 top left) to our sagged style (Figure 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 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 10. We stress that Figure 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

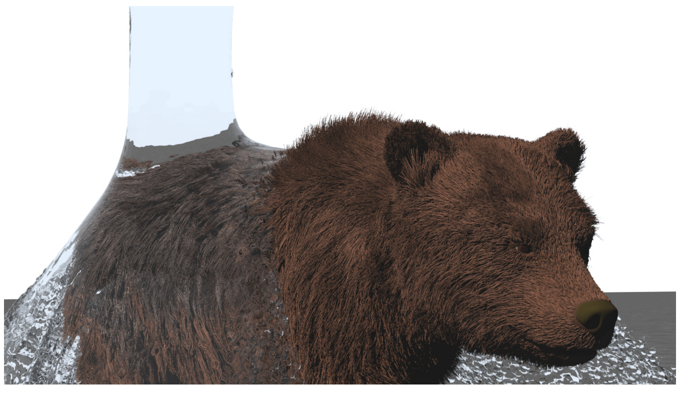


Fig. 13. Water poured over creature.

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 11 (top). Then, if desired, simulate a tetrahedral mesh bound to the KDSM, potentially affected by water drag, etc., as in Figure 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 4 and 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.

3.4 KDSM for Hair-Water Interaction

Motivated by [10] (see also the follow-up work of [48]), 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 [10] 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,

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 1
List of examples with their resolutions and average runtime per frame. Note that the first layer, generating the KDSM, is not included.

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 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 12 and 13 (see also Figure 2) 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 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

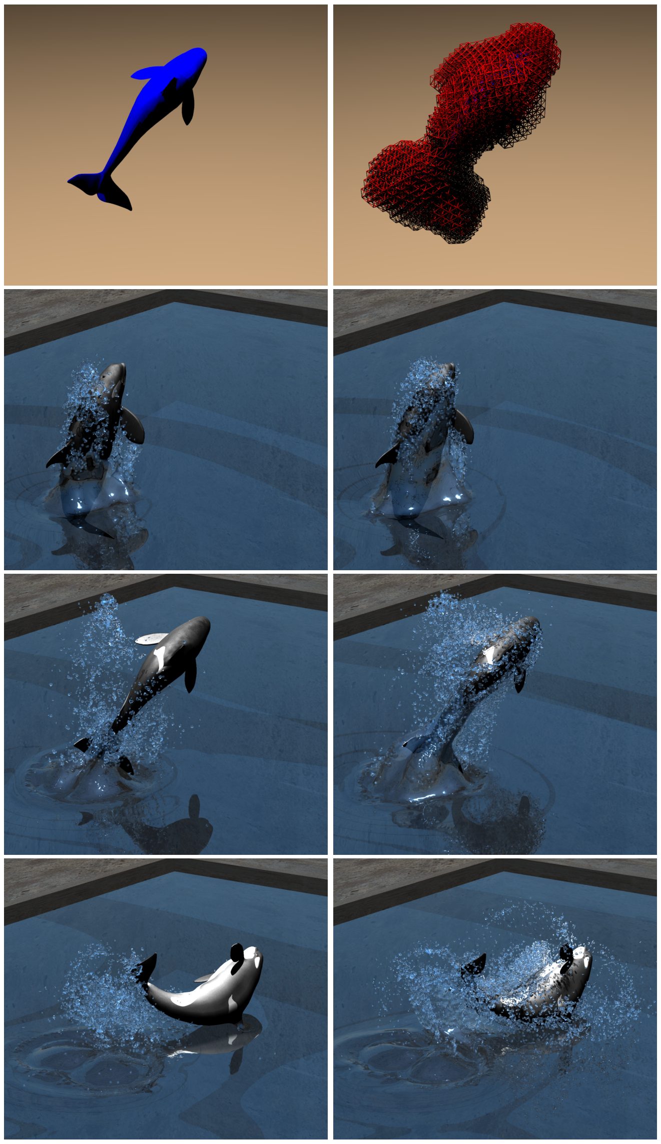


Fig. 14. (Top) Creature geometry and KDSM. (Left) No KDSM. (Right) Same frames as on the left, except using KDSM for increased drag on the water from the creature.

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 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.

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 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 4 and 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 [19], 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 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 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 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 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 4 and 13) use dynamic skinning.The bear water example (Figure 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 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 1.

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.

Acknowledgements

The authors would like to acknowledge Ed Quigley for helping us with writing and compositing videos, and Winnie Lin for voice acting. Research supported in part by ONR N00014-13-1-0346, ONR N00014-17-1-2174, ARL AHPCRC W911NF-07-0027, and NSF CNS1409847. M.L. was supported by Samsung Scholarship and MPC-VCC Summer Scholarship. D.H. was supported by NDSEGF. M.B. was supported by VMWare Fellowship. Computing resources were provided in part by ONR N00014-05-1-0479.

References

[1] J. T. Kajiya and T. L. Kay, "Rendering fur with three dimensional textures," in *Comput. Graph. (Proc. ACM SIGGRAPH 90)*. ACM, 1989, pp. 271–280.

[2] R. E. Rosenblum, W. E. Carlson, and E. Tripp, "Simulating the structure and dynamics of human hair: modelling, rendering and animation," *J. Vis. and Comput. Anim.*, vol. 2, no. 4, pp. 141–148, 1991.

[3] K. Anjyo, Y. Usami, and T. Kurihara, "A simple method for extracting the natural beauty of hair," in *Comput. Graph. (Proc. ACM SIGGRAPH 92)*, vol. 26. ACM, 1992, pp. 111–120.

[4] L. Petrovic, M. Henne, and J. Anderson, "Volumetric methods for simulation and rendering of hair," *Pixar Anim. Studios*, vol. 2, no. 4, 2005.

[5] K. Ward, M. Simmons, A. Milne, H. Yosumi, X. Zhao, and W. D. A. Studios, "Simulating rapunzel's hair in disney's tangled." in *SIGGRAPH Talks*, 2010.

[6] M. Simmons, K. Ward, H. Yosumi, H. Leo, and X. Zhao, "Directing hair motion on tangled," in *ACM SIGGRAPH 2011 Talks*. ACM, 2011, p. 41.

[7] H. Iben, M. Meyer, L. Petrovic, O. Soares, J. Anderson, and A. Witkin, "Artistic simulation of curly hair," in *Proc. of the 12th ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* ACM, 2013, pp. 63–71.

[8] K. Wilson, A. McAdams, H. Leo, M. Simmons, and W. D. A. Studios, "Simulating wind effects on cloth and hair in Disney's Frozen." in *SIGGRAPH Talks*, 2014, pp. 48–1.

[9] M. Simmons and B. Whited, "Disney's hair pipeline: crafting hair styles from design to motion," in *Eurographics 2014 Industrial Presentations*, 2014.

[10] W. Rungjiratananon, Y. Kanamori, and T. Nishita, "Wetting effects in hair simulation," in *Comput. Graph. Forum*, vol. 31, no. 7. Wiley Online Library, 2012, pp. 1993–2002.

[11] S. Hadap and N. Magnenat-Thalmann, "Modeling dynamic hair as a continuum," *Comput. Graph. Forum*, vol. 20, no. 3, 2001.

[12] Y. Bando, B.-Y. Chen, and T. Nishita, "Animating hair with loosely connected particles," in *Comput. Graph. Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 411–418.

[13] E. Plante, M.-P. Cani, and P. Poulin, "Capturing the complexity of hair motion," *Graph. Models*, vol. 64, no. 1, pp. 40–58, 2002.

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

[15] K. Ward, M. C. Lin, J. Lee, S. Fisher, and D. Macri, "Modeling hair using level-of-detail representations," in *Proc. of Comput. Anim. and Social Agents (CASA)*, 2003, p. 41.

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

[17] M. Chai, C. Zheng, and K. Zhou, "A reduced model for interactive hairs," *ACM Trans. Graph.*, vol. 33, no. 4, p. 124, 2014.

[18] ——, "Adaptive skinning for interactive hair-solid simulation," *TVCG*, 2016.

[19] A. Selle, M. Lentine, and R. Fedkiw, "A mass spring model for hair simulation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 27, no. 3, pp. 64.1–64.11, Aug. 2008.

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

[21] G. Daviet, F. Bertails-Descoubes, and L. Boissieux, "A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics," in *ACM Trans. Graph.*, vol. 30, no. 6. ACM, 2011, p. 139.

[22] F. Bertails-Descoubes, F. Cadoux, G. Daviet, and V. Acary, "A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies," *ACM Trans. Graph.*, vol. 30, no. 1, p. 6, 2011.

[23] D. M. Kaufman, R. Tamstorf, B. Smith, J.-M. Aubry, and E. Grinspun, "Adaptive nonlinearity for collisions in complex rod assemblies," *ACM Trans. Graph.*, vol. 33, no. 4, p. 123, 2014.

[24] E. Sifakis, S. Marino, and J. Teran, "Globally coupled collision handling using volume preserving impulses," in *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* Eurographics Association, 2008, pp. 147–153.

[25] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, "Air meshes for robust collision handling," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 133:1–133:9, July 2015. [Online]. Available: http://doi.acm.org/10.1145/2766907

[26] K. Wu and C. Yuksel, "Real-time hair mesh simulation," in *Proc. of the 20th ACM SIGGRAPH Symp. on Interactive 3D Graph. and Games*. ACM, 2016, pp. 59–64.

[27] P. Volino and N. Magnenat-Thalmann, "Animating complex hairstyles in real-time," in *Proc. of the ACM Symp. on Virt. Reality Software and Tech.* ACM, 2004, pp. 41–48.

[28] ——, "Real-time animation of complex hairstyles," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 12, no. 2, pp. 131–142, 2006.

[29] U. Bonanni, M. Montagnol, and N. Magnenat-Thalmann, "Multilayered visuo-haptic hair simulation," *The Vis. Comput.*, vol. 24, no. 10, pp. 901–910, 2008.

[30] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, 2008, pp. 1–6.

[31] M. Müller and N. Chentanez, "Wrinkle meshes," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 2010, pp. 85–92.

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

[33] D. Ali-Hamadi, T. Liu, B. Gilles, L. Kavan, F. Faure, O. Palombi, and M.-P. Cani, "Anatomy transfer," in *ACM SIGGRAPH Asia 2013 papers*, ser. SIGGRAPH ASIA '13, 2013, pp. 188:1–188:8.

[34] 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.*, 2015, pp. 175–183.

[35] H. Xu and J. Barbič, "Pose-space subspace dynamics," *ACM Trans. Graph.*, vol. 35, no. 4, p. 35, 2016.

[36] 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*, 2003, pp. 103–114.

[37] M. Müller, T.-Y. Kim, and N. Chentanez, "Fast simulation of inextensible hair and fur," *VRIPHYS*, vol. 12, pp. 39–44, 2012.

[38] 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.

[39] 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.*, 2003, pp. 28–36.

[40] L. Hu, C. Ma, L. Luo, and H. Li, "Single-view hair modeling using a hairstyle database," *ACM Transactions on Graphics (Proceedings SIGGRAPH 2015)*, vol. 34, no. 4, July 2015.

[41] S. D. Bowline and Z. Kacic-Alesic, "A unified dynamics pipeline for hair, cloth, and flesh in Rango," in *ACM SIGGRAPH 2011 Talks*. ACM, 2011.

[42] R. Weinstein, E. Guendelman, and R. Fedkiw, "Impulse-based control of joints and muscles," *IEEE Trans. on Vis. and Comput. Graph.*, vol. 14, no. 1, pp. 37–46, 2008.

[43] A. Bruderlin, "A method to generate wet and broken-up animal fur," in *Comput. Graph. and App., 1999. Proc. Seventh Pacific Conf. on*. IEEE, 1999, pp. 242–249.

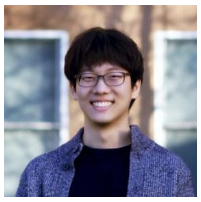
[44] K. Ward, N. Galoppo, and M. C. Lin, "Modeling hair influenced by water and styling products," in *Proc. of Comput. Anim. and Social Agents (CASA)*, 2004, pp. 207–214.

[45] ——, "Simulating and rendering wet hair," in *SIGGRAPH 2004 Sketches*. ACM Press, 2004, p. 42.

[46] K. Ward, N. Galoppo, and M. Lin, "Interactive virtual hair salon," *Presence: Teleoperators and Virt. Env.*, vol. 16, no. 3, pp. 237–251, 2007.

[47] R. Gupta and N. Magnenat-Thalmann, "Interactive rendering of optical effects in wet hair," in *Proc. of the 2007 ACM Symp. on Virt. Reality Software and Tech.* ACM, 2007, pp. 133–140.

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



Minjae Lee received his B.S. in Computer Science minoring in Art from Carnegie Mellon University in 2012. He is currently pursuing his Ph.D. at Stanford University where he is supported by Samsung Scholarship and MPC-VCC Summer Scholarship. His research interests include computer graphics, physically based simulation, computational biochemistry, and games.

PLACE PHOTO HERE

David Hyde photography and biography not available at the time of publication.

Michael Bao received his B.S. in Computer Science from UC Berkeley. He is currently working at Industrial Light + Magic and pursuing his Ph.D. in the Department of Computer Science at Stanford University where he is supported by VMWare Fellowship. His research interest is in using both physically based simulation and machine learning to create high quality facial animations for visual effects.

Ronald Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded an Academy Award from The Academy of Motion Picture Arts and Sciences, the National Academy of Science Award for Initiatives in Research, a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), a Sloan Research Fellowship, the ACM Siggraph Significant New Researcher Award, an Office of Naval Research Young Investigator Program Award (ONR YIP), the Okawa Foundation Research Grant, the Robert Bosch Faculty Scholarship, the Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. He has served on the editorial board of the Journal of Computational Physics, Journal of Scientific Computing, SIAM Journal on Imaging Sciences, and Communications in Mathematical Sciences, and he participates in the reviewing process of a number of journals and funding agencies. He has published over 110 research papers in computational physics, computer graphics, and vision, as well as a book on level set methods. For the past fifteen years, he has been a consultant with Industrial Light + Magic. He received screen credits for his work on "Terminator 3: Rise of the Machines," "Star Wars: Episode III – Revenge of the Sith," "Poseidon," and "Evan Almighty."