A Framework for Rendering Complex Multi-Scattering Effects on Hair

Xuan Yu* University of Delaware
Jason C. Yang† Advanced Micro Devices
Justin Hensley‡ Advanced Micro Devices
Takahiro Harada§ Advanced Micro Devices
Jingyi Yu¶ University of Delaware

Figure 1: *Our solution handles various multi-scattering effects in hair rendering under a unified framework. From left to right, we render a complex hair model with 114K hair strands and 6.8MK line segments under single scatting(anti-aliased), volumetric shadows (VS) and volumetric shadows plus translucency and anti-aliasing On an AMD Radeon 5800 graphics card, our framework renders over 15 fps.*

Abstract

The appearance of hair plays a critical role in synthesizing realistic looking human avatars. However, due to the high complexity in hair geometry and the scattering nature of hair fibers, rendering hair with photorealistic quality and at interactive speeds remains as an open problem in computer graphics. Previous approaches attempt to simplify the scattering model to only tackle a specific aspect of the scattering effects. In this paper, we present a new approach to simultaneously render multiple scattering effects including volumetric shadows, translucency, and anti-aliasing under a unified framework. Our solution uses a shadow-ray path to produce volumetric self-shadows and an additional view-ray path to produce translucency. To compute and accumulate the contribution of individual hair fibers along each (shadow or view) path, we develop a new GPU-based K-Buffer technique that can efficiently locate the K nearest scattering locations and combine them in the correct order. Compared with existing multi-layer based approaches[Kim and Neumann 2001; Yuksel and Keyser 2008; Sintorn and Assarsson 2009], we show that our K-Buffer solution can more accurately reproduce the shadowing and translucency effects. Further, we present an anti-aliasing scheme that directly builds upon the K-Buffer. We implement all three effects (volumetric shadows, translucency, and anti-aliasing) under a unified rendering pipeline. Experiments on complex hair models demonstrate that our new solution produces near photorealistic hair rendering at very interactive speed.

Keywords: Hair rendering, Scattering, GPU, Anti-aliasing

*e-mail: xuanyu@udel.edu
†e-mail:jasonc.yang@amd.com
‡e-mail:justin.hensley@amd.com
§e-mail:takahiro.harada@amd.com
¶e-mail:yu@eecis.udel.edu

Links: DL PDF

1 Introduction

The appearance of hair, such as its translucency, complexity, styling, etc, plays a critical role in producing realistic-looking human avatars in computer games and feature animations [Ducheneaut et al. 2009]. In recent years, tremendous efforts have been focused on developing systems and tools for acquiring real 3D hair geometry and appearance models [Yuksel et al. 2009; Paris et al. 2004; Paris et al. 2008; Kim and Neumann 2002; Ward et al. 2007] that can later be used on virtual characters. Rendering hair at photo-realistic quality, however, still mainly relies on offline approaches such as ray tracing due to the complexity of hair geometry and the the scattering nature of hair fibers. For example, a real human hair model contains around one hundred thousand hair fibers on average and the hair fibers exhibit scattering effects.

Existing interactive hair rendering schemes attempt to simplify the scattering model to only tackle a specific type of the scattering effects such as volumetric shadows[Kim and Neumann 2001; Zinke et al. 2008; Yuksel and Keyser 2008; Bertails et al. 2005].

Yuksel et al. proposed to use the Deep Opacity Map(DOM) algorithm to account for complex shadowing effects between hair fibers. Zinke et al.[Zinke et al. 2008] derived a simplified physical model to approximate light propagations inside the hair volume. However, these methods do not account for light scattering from the hair to the viewpoint, without which the results can appear artificial or even plastic looking (Fig.10(b)). The recent work by Sintern et al.[Sintorn and Assarsson 2009] is probably the only technique that approximates both the shadowing and the translucency effects. Their technique, however, uniformly blends hair fragments along each ray without considering their order. Another understudied problem in hair rendering is anti-aliasing, a crucial step to guarantee the smooth-looking appearance as shown in Fig.5(a). The state-of-the-art solution is to use hardware-based multi-sampling[Kim and Neumann 2001; Yuksel and Keyser 2008]. However, for thin hair fibers, a large number of samples are required to reduce the aliasing artifacts, which would significantly degrade the performance.

In this paper, we present a new approach to simultaneously render various scattering effects including volumetric shadows, translucency, and anti-aliasing on complex hair models. Our solution uses a shadow-ray path similar to [Zinke et al. 2008] to produce volumetric self-shadows. We also use an additional view-ray path to model light propagations from the hair volume to the eye for producing translucency. To compute and accumulate the contribution of individual hair fibers along each ray (shadow or view) path, we develop a new GPU-based K-Buffer solution. A K-Buffer[Bavoil et al. 2007] stores the foremost K fragments along each ray. For hair rendering, we show that a K-Buffer allows us to locate the critical scattering events along the ray and combine them in the correct order. Compared with existing multi-layer based approaches[Kim and Neumann 2001; Yuksel and Keyser 2008; Sintorn and Assarsson 2009], we show that our K-Buffer solution can more accurately reproduce the shadowing and translucency effects. Further, we present an anti-aliasing scheme that directly builds upon the K-Buffer. We use a geometry shader to ensure thin hair fibers will be properly rasterized. We then estimate their contributions towards each pixel and use the K-Buffer to combine them. We implement all three effects (volumetric shadowing, translucency, and anti-aliasing) under a unified rendering pipeline. Experiments on complex hair models demonstrate that our new solution produces near photorealistic hair rendering at very interactive speed.

2 Related Work

Rendering realistic-looking hair is a challenging task in computer graphics. Recent work[Moon and Marschner 2006; Moon et al. 2008; Zinke et al. 2004; Zinke et al. 2008] suggests that the key is to faithfully model and then render the multiple-fiber scattering effects within the hair volume. Accurate simulation of multiple scattering effect generally requires high expensive off-line approaches such as Monte-Carlo path tracing. Even with accelerated approach such as spherical harmonics[Moon and Marschner 2006] or efficient photon mapping[Moon et al. 2008], it still takes hours to render one frame.

To render the multi-scattering effects on hair at interactive speeds, recent approaches attempt to model two specific phenomenon caused by multi-scattering: 1) volumetric self-shadowing caused by complex occlusions between the hair fibers and 2) the translucent appearance due to the scattering nature of the hair material.

Volumetric Shadows. On the shadow side, a well received technique is the Deep Shadow Map (DSM)[Lokovic and Veach 2000]. Unlike traditional shadow map techniques that only store a single depth value at each pixel, DSM stores at each pixel a visibility(or transmittance) function that is generated by first sampling and then compressing the visibility values at various depths along the ray. DSM is able to produce high quality volumetric shadows. However, its sampling and compression process can be quite expensive and therefore DSM is not intended for real time applications.

Opacity Shadow Map[Kim and Neumann 2001]simplified the visibility function sampling process by dividing the scene into several depth slices where each slice is rendered separately with its fragments additively blended onto an alpha map. The resulting alpha maps will then be used to approximate the visibility function. OSM fits well onto the GPU via multi-pass rendering. However, for complex geometry such as hair fibers, 40-80 layers of alpha maps are generally required to produce high quality, smooth shadowing result, and the overhead between the passes can greatly reduce the performance.

More recently, Yuksel and Keyser[Yuksel and Keyser 2008] proposed a Deep Opacity Map technique that slices the scene into fewer layers. The main idea there is to use the curved depth layers instead of the planar ones to more effectively approximate hair geometry. Using the DOM, one can reduce the number of layers. However, their rendering quality relies on choosing the proper layers to avoid rendering artifacts. We in contrast builds upon K-Buffer that is able to produce more accurate volumetric shadows with less samples.

Translucency. The cause of the translucent appearance of hair is mainly due to light transmittance through the hair fiber. It is important to note that hair translucency is inherently more difficult. Shadows are a ”grey-scale level” effect as they only cause intensity changes that are less discernable to human eyes. In contrast, transparency is a ”color level” effect as it combines the color of nearby fragments and therefore requires using much more accurate and robust techniques to minimize the visual artifacts. Similar to shadows, however, the key challenge in producing translucency is to determine the hair fibers and more importantly their orders along a light ray.

The state-of-the-art in hair rendering includes the quick GPU sorting algorithm by Sintorn at al.[Sintorn and Assarsson 2008] that aims to first sort the hair fibers and then alpha blend them. Their original approach only computes the ordering at the geometry level rather than at the pixel level, and hence is less correct. To improve accuracy, they have recently developed an Occupancy Map (OM) algorithm[Sintorn and Assarsson 2009] that combines a high resolution binary occupancy map with a low resolution alpha slab map. Their method can also be used to render more accurate hair shadowing effects. However, similar to all multi-layer approximation approaches, OM requires approximating scene geometry into predefined layers. In the case of hair rendering, since the spatial distribution of the hair fibers is generally unpredictable and varies significantly across the pixels, its approximation can yield poor result when the sliced layers do not lie close to the actual hair geometry.

It is also possible to view shadows and translucency using a unified model. A notable stream of work in realistic hair rendering is the dual scattering approximation approaches proposed by Zinke et al.[Zinke et al. 2008]. Their technique generalized the scattering effect along the shadow path and extend the shadow map to a forward scattering map based on physical models. In their model, however, translucency is approximated via the backward scattering term that is modeled as a geometry-independent material property. As a result, their technique does not model the complete cause of translucency, e.g., it did not consider how light propagates towards the eye. Our technique completes their model by uniformly model the path from the light source towards the hair and from the hair to

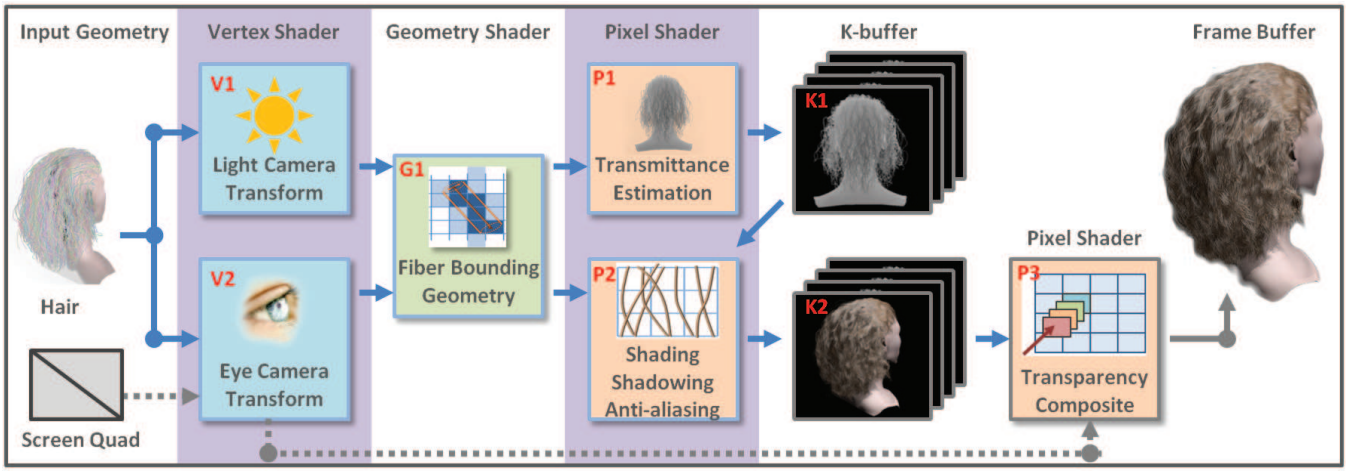


Figure 2: *A diagram showing the pipeline of our hair rendering framework. Our solution combines programmable vertex, geometry, and pixel shaders with a new GPU-based linked list (K-Buffer).*

towards the eye.

K-Buffer. Finally, our approach benefits from recent advances on supporting complex data structures on the GPU, in particular, the K-Buffer proposed by Bavoil et al.[Bavoil et al. 2007]. K buffer generalizes the tradition z-buffer by storing the foremost k fragments rather than closet one at each pixel and hence can be potentially used for rendering traditionally challenging effects. In particular, K-Buffer would be a suitable solution for faithfully reproducing translucency. For example, if we assume a hair fiber has a transmittance of 80 percent, a K-Buffer of 16 layers will contain 97 percent of the energy carried by the incident light ray.

The original K-Buffer implementation on GPU, however has hazards due to the limitation on the graphics hardware. Most recently, Yang et al.[Yang et al. 2010] proposed an A-buffer implementation based on the newly feature on the graphics card. An a-buffer differs from k-buffer in that it stores all fragments along a ray at its corresponding pixel. Conceptually, as long as the depth values are stored within the A-buffer, one can build a K-Buffer at a later pass. We designed an efficient K-Buffer construction stage for rendering layer-intensive geometry such as hair, it allows us to accurately simulate shadowing and translucency effects at near real-time speed.

3 Our Approach

In this section, we present our rendering algorithm and its implementation using the latest features on the graphics hardware. Fig.2 shows the pipeline of our algorithm.

Before proceeding, we explain our notation. Similar to previous approaches[Marschner et al. 2003; Zinke et al. 2008], we model each hair strand as a one dimensional fiber with central axis u. Further, we assume distant lighting, i.e., all incident light rays have an identical direction of ω_d and radiance of $L_d(\omega_d)$. We use the normal plane to u to define the azimuthal angle ϕ and the longitudinal angle θ. Each direction ω can then be described using ϕ and θ as shown in Fig.3

For each point x on the fiber, we use $L_i(x, \omega_i)$ and $L_o(x, \omega_o)$ to describe the incident and outgoing radiance from direction ω_i and towards ω_o, respectively.

Our goal is to compute the radiance L_p at each pixel p in the image. To do this we compute its corresponding ray r_p of direction w_p. Due to complex hair geometry, r_p is likely to hit multiple fibers

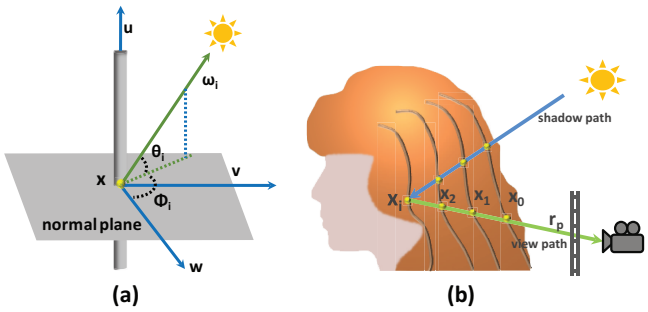


Figure 3: *Our Multi-Scattering Model. Left: our notation of angles and directions. Right: we model volumetric shadows via the shadow scattering path and translucency via the view scattering path.*

as shown in Fig.3(b). Therefore, L_p should be computed by accumulating radiances from multiple hair fibers x_i as:

$$L_p = \sum_{i=0}^{n} L_o(x_i, \omega_p) \cdot T(x_i, \omega_p) \quad (1)$$

where $T(x_i, \omega_o)$ is the transmittance function that accounts for scattering from the hair towards the eye.

Our main tasks are to 1) effectively approximate how light scatters within the hair volume, i.e., $L_i(x, \omega_i)$ and 2) to accumulate the contributions of different fragments towards the eye, i.e., $L_o(x, \omega_o)$.

3.1 Light-to-Hair: Self Shadowing Effects

Under the directional light assumption, the incident radiance from the light to hair can be simplified as:

$$L_i(x, \omega_i) = L_d(w_d)\Psi(x, w_d, w_i) \quad (2)$$

where Ψ is the multiple scattering function that describes the fraction of lights from the light source that is scattered inside the hair volume and finally arrives at point x from direction ω_i.

Accurate evaluation of multiple scattering Ψ generally requires Monte-Carlo methods. Marschner et al.[Marschner et al. 2003] have shown that the scattering of hair fiber has a very large forward scattering component. Therefore, one can approximate

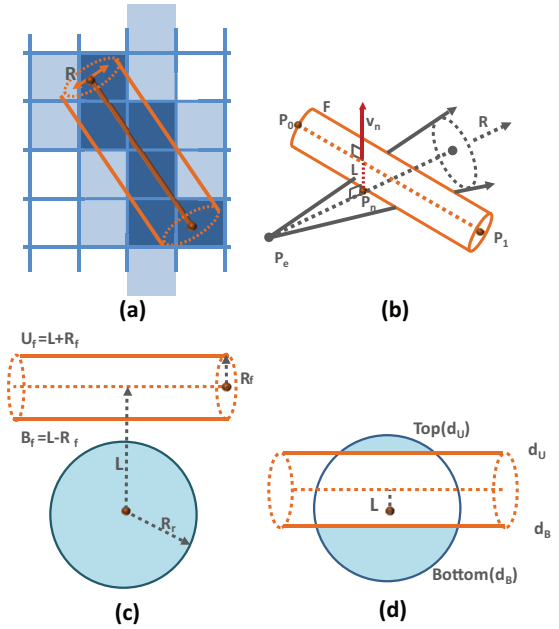


Figure 4: *Our anti-aliasing solution. (a) Without anti-aliasing, a thin fiber will map to jig-jagged edges (dark blue). (b) We compute partial coverage of a fiber $F(p_0, p_1)$ on a ray R. (c) show the case when F lies completely outside of the pixel cone and (d) shows when F lies partially inside the cone.*

the Ψ function in terms of the forward scattering events along the shadow path[Zinke et al. 2008]. Like most hair shadowing techniques[Kim and Neumann 2001; Lokovic and Veach 2000; Yuksel and Keyser 2008], we assume $\Psi_l \approx T_l(x, w_d)$ where $T_l(x, w_d)$ is the transmittance function at fiber point x along the light direction ω_d. We choose not to include another Gaussian spread term as suggested in [Zinke et al. 2008] as $T_l(x, w_d)$ already contains the scattering energy distribution. The light transmittance at fiber x is a result of all the scattering events along the shadow path as shown in Fig.3(b).

At each scattering event, the amount of light penetration into a fiber x_i is determined by the attenuation factor $A(\eta, \theta_i)$[Marschner et al. 2003]. $A(\eta, \theta_i)$ is calculated from Fresnel formulas using the incident angle θ_i and refraction index η. Evaluating $A(\eta, \theta_i)$ at each scattering event along the shadow path gives a reliable estimation of $T_l(x, \omega_d)$ if the shadow beam is thin enough (compare with hair fiber) and is densely sampled. In shadow map implementation, it is equivalent to evaluating $A(\eta, \theta_i)$ at each shadow pixel.

As is shown in Fig.4(a), hair fibers in many cases only partially cover a single pixel. To handle this issue, we introduce another term $C(x_i, \omega_i)$ to describe the spatial coverage of the fiber within the shadow ray cone. We conduct a ray cone-fiber intersection to compute the spatial coverage term as illustrated in Fig.4(d). Using the coverage term, we have the amount of penetrating light at each captured event as:

$$A(\eta, \theta_i) \cdot C(x_i, \omega_i) + (1 - C(x_i, \omega_i))$$

Substituting T_l into Eqn. (3), we have:

$$L_i(x, \omega_i) \approx Ld(w_d) \prod_{i=1}^{n} [A(\eta, \theta_i) C(x_i, \omega_i) + (1 - C(x_i, \omega_i))] \tag{3}$$

In essence, it states that we can estimate the incident light of a fiber x as a product of the n scattering events along its shadow path. In our implementation, this step is done at pipeline stage P2 in Fig.2. The result corresponds to the volumetric self-shadow effects shown in Fig.10(2nd collum). The n scattering events required for estimating transmittance is kept in the volume shadow map using our K-buffer that would be generated at pipeline stage K1. The attenuation factor $A(\eta, \theta_i)$ and the coverage term $C(x_i, \omega_i)$ are calculated when generating the shadow map at pipeline stage P1 shown in Fig.2.

3.2 Hair-to-Eye: Transparency

Once we estimate the incident radiance L_i from the the light to the hair, we can compute hair shading and then hair-to-eye scattering.

We use Marschner's model for approximating hair BSDF function ω:

$$L_o(x, w_o) = L_i(x, w_i) \dot{S}(w_i, w_o) \cos(\theta_i) \tag{4}$$

The shading results in a narrow lobe of primary highlights that reflect the light color as well as a wide lobe of second highlights that show the color of the hair itself and a forward scattering component that contributes to the our transmittance function.

To compute the radiance L_p at each pixel p, we substitute Eqn. (4) into Eqn. (1). Notice that our assumption here is that only points lying on ray r_p contribute to L_p.

$$L_p = \sum_{i=0}^{n} L_i(x, w_i) \dot{S}(w_i, w_o) \cos(\theta_i) \cdot T(x_i, \omega_p) \tag{5}$$

The summation of the contributions of all fibers along the ray r_p effectively reproduces translucent-looking hair as shown in Fig.10(3rd collum). The summation is carried out in pipeline stage P3 in Fig.2. The transmittance function $T(x_i, \omega_p)$ here is defined in a similar way as T_l with an attenuation factor and a coverage term. Again, the coverage term is particularly important because it not only provides accurate transmittance for shading but also helps create smooth anti-aliased fibers as shown in Fig.5(a).

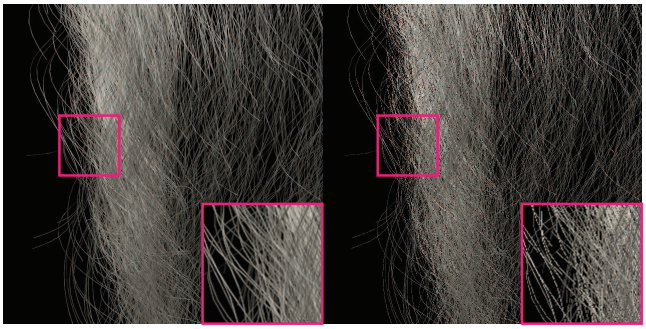
4 Anti-Aliasing

A unique feature of our hair rendering framework is that it directly supports anti-aliasing. Computer synthesized images exhibit aliasing artifacts due to discrete sampling. In the case of hair rendering, aliasing can be particularly sever when rendering thin primitives (fibers) that map to lots of edge pixels as shown in Fig.5(b).

4.1 View Space Expansion

Our solution for anti-aliasing is a two-stage process: we first estimate a coverage value for each rasterized fiber fragment in the pixel shader and pass it down to the pipeline; at every pixel, we then compose K nearest partially covered fragments to achieve anti-aliasing. It is important to note that the partial fragments have to be composed in depth order which is not supported by the traditional Z-Buffer.

Recall that in the rasterization process [Microsoft 2010], a triangle will be rasterized via the pixel center test (or via diamond test if fibers are rendered using line primitive). A partial fragment (i.e., the one that is unable to cover the pixel center) will fail the test and hence would not be rasterized. As a result, the fragment would not exist in the pixel shader for coverage estimation and composition. Re resolve this issue, we add an additional geometry shader that expand each input fiber (represented as a line segment) to a



(a) (b)

Figure 5: *Comparisons of the hair rendering results with and without anti-aliasing. More results can be found in the supplementary video.*

bounding volume so that it would be large enough to cover all its potentially touching pixels. To do so, we first expand the input line segment to a quad whose size is determined by the radius of fiber as shown in Fig.4(a). We then apply the view-projection transform to map each expanded fiber onto the 2D image plane. We extend the edge outwards by 2 pixels to ensure no partial fibers would be missed (light blue pixels in Fig.4(a). Finally, to estimate the coverage value of each fiber within the ray cone in pixel shader, we pass the two vertices of the expanded fiber down. Notice the fragments that have zero coverage values will not be sent it to later stages of the pipeline and therefore our anti-aliasing scheme will not incur unnecessary overhead.

4.2 Computing Coverage Percentage

Next, we compute the coverage of the fiber on each pixel. We assume that each pixel on the image maps to a cone of rays or a ray cone (Fig.4(b)). Given an eye ray R and a hair fiber F defined by two endpoints $\dot{p_0}$ and $\dot{p_1}$, we can first approximate the "pseudo depth" d of F with respect to the viewpoint $\dot{p_e}$. To do so, we compute the shortest distance L between R and F and its corresponding vertex $\dot{p_n}$ along R as shown in Fig.4(b).

The radius of the pixel cone at point $\dot{p_n}$ can be computed as:

$$R_r = 2|\dot{p_e} - \dot{p_n}|\frac{tan(\alpha/2)}{ResY} \quad (6)$$

where α is the vertical field-of-view of the camera and $ResY$ is the Y directional resolution.

Our goal is to compute the percentage of pixel cone area that is covered by fiber F as shown in Fig.4(b). Assume that the radius of the fiber F is R_f and p_n is the origin of the coordinate system, we need to handle two cases:

1) F lies completely outside the circle. Specifically, we can compute the upper and the bottom bound U_f and B_f of the fiber as $U_f = L + R_f$ and $B_f = L - R_f$. If $B_f > R_r$ or $U_f < -R_r$, we can conclude that the fiber lies completely outside the pixel.

2) In the more general case that the fiber partially covers the circle, we can compute $d_U = min(R_r, U_f)$ and $d_B = max(-R_r, B_f)$ and then compute the area covered between d_U and d_B of the circle as $\Pi R_r^2 - Top(d_U) - Bottom(d_B)$ where $Top(d_U)$ is the area of the top crown of the circle formed at d_U and $Bottom(d_B)$ is the area of the bottom crown formed at d_B, as shown in Fig.4(d).

5 K-Buffer Construction

The core component of our solution Fig.2 is to locate the scattering events and accumulate their contributions using K-Buffer. In this section, we present two solutions based on the recently proposed GPU Linked List. The first is a two-pass algorithm based on the A-Buffer [Yang et al. 2010] and the second is a single pass algorithm that directly creates the K-Buffer.

5.1 A Two-Pass Algorithm

The idea of concurrent linked list or A-Buffer construction on the GPU was recently proposed in [Yang et al. 2010], based on the latest hardware features: append buffer and atomic operations. We refer the readers to [Yang et al. 2010] for detailed implementations of the A-Buffer. Our goal here is to construct the hair volume transmittance function using the linked list through a two pass algorithms: in the first pass the per-pixel linked lists are constructed to capture all fragments (i.e, building a A-Buffer) along each shadow or view ray path; in the second pass, we reconstruct the transmittance functions using ordered fragments in the A-Buffer. However, since hair geometry is usually highly complex, each pixel can contain a few hundreds of fragments. Therefore, ordering the fragments via naive sorting (as was implemented in the original A-Buffer paper [Yang et al. 2010]) would be too slow to achieve real-time performance.

Recall that when a light ray travels within the hair volume, its energy decreases after each scattering event. If we assume 25 percent decay rate, only 1 percent energy would remain after the first 16 scattering events. This indicates that we only need to care about the first few (K) scattering events along each ray path. Our solution hence is to keep the store the K foremost but unsorted elements in the list. This data structure is often referred to as the K-Buffer. The order of these element will be determined later via a much smaller (of size K) loop-based scheme.

To construct the K-Buffer from the A-Buffer, we store the K ($K < 64$ on an ATI Radeon 5800) foremost elements in register space to allow fast selection and replacement. We transverse all elements in the A-Buffer and when a new fragment A is picked, we select the farthest fragment B from the K-buffer and compare/replace B with A. This can be done using an unrolled loop followed by a conditioned replacement operation. It is worthy noting that Salvi et al. [Salvi et al. 2010] also select a subset of nodes from the A-Buffer via a heuristic, adaptive approach (e.g., the min area criteria) for constructing the transmittance function. However, their selection scheme requires more operations than our fast selection and replacement and their selection criteria may not guarantee the optimal approximation.

5.2 A Single Pass Algorithm

The main limitation of the previous algorithm is that the A-Buffer and the K-Buffer need to be conducted via two separate passes. Since we are only interested in the K foremost elements, constructing the A-Buffer seems wasteful and unnecessary. Conceptually, a straightforward solution is to directly build the K-Buffer. We hence present a new, single-pass solution based on the latest graphics hardware.

We initiate each K-sized linked list (for a screen pixel) to have a head node with minimum depth (zero) value and a tail node with the maximal possible depth value (one). When a hair fiber is rasterized, its fragment(s) will traverse the corresponding linked list and get inserted at the proper location so that the first K elements in the list will still remain in order (from min to max).

Recall that inserting a new fragment A_0 to the linked list consists of the multiple steps: 1)locate the two nodes F_i and F_{i+1} where A should be inserted in between, 2)allocate node A_0 and 3)change the successor of A_0 and F_i. In order to correctly conduct the insertion

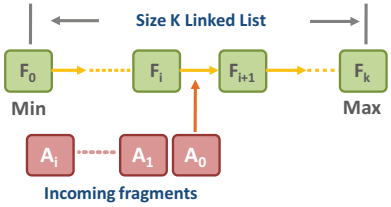


Figure 6: *When a K-Buffer is being constructed, multiple fragments can simultaneously request to be inserted at the same location. To avoid write conflict and to guarantee serializability, we have designed a special GPU technique (Section 5.2)*

operation in parallel, step 3) needs be atomic since multiple fragments (running in parallel) A_1 to A_k may also be inserted to the same location at the same time. However, an atomic step 3) alone is still insufficient to guarantee correct insertion. For example, consider two fragments A_0 and A_1 are both being inserted between F_i and F_{i+1} and A_0 gets inserted first. It is important to note that to A_1 its originally detected successor F_{i+1} now becomes A_0 which can be closer to A_1. As a result, we should no longer insert A_1 at this location. To resolve this issue, we hence add an additional *verification* step to 3), i.e., we verify that the successor right before insertion is the same as the first detected one.

In our implementation we use the DirectX 11 [Microsoft 2010] atomic function:

$InterlockedCompareExchange($ *in dest*, *in compareValue*, *in value*, *out originalValue*$)$;
The function atomically compares the value in $dest$ to $compareValue$. It then stores $value$ in $dest$ and returns this new value if they match or returns the original value in $dest$ if they do not. In our case, we call

$InterlockedCompareExchange(F_i.Next, \quad address(F_{i+1}), address(A), oldAddress)$

and then compare $oldAddress$ with $address(F_{i+1})$ to see if the insertion is successful. If they match, the fragment is then correctly inserted and we can then break the loop and end the pixel shader. If not, we continue traversing the linked list from F_i and look for the next potential location for insertion.

6 Results and Discussion

Our algorithm is tested on AMD radeon 5800 GPU with 2G video memory. We use DirectX 11 Shader Model 5.0 as graphics API. The K-Buffer algorithms for computing volumetric shadow and translucency are implemented as the last stage of our rendering pass using a pixel shader, by selecting and sending the foremost K fragments to a pre-allocated read-write global memory space.

6.1 K-Buffer vs. A-Buffer

The speed of our GPU K-Buffer construction is the most important component of our rendering pipeline as both the shadow path and view path transmittance estimations are implemented based on the K-Buffer. Since our solution targets at rendering highly complex hair geometry, hundreds of small hair fibers will be rasterized onto each pixel. Our two-pass K-Buffer construction algorithm is able to achieve real-time performance as shown in Fig.9. In particular, we compare our K-Buffer construction algorithm with the Order Independent Transparency (OIT) scheme [Yang et al. 2010] that applies a selection sorting to all fragments in the A-Buffer and composite the foremost K fragments. In our experiment, we render a hair model with 50K strands, 2.4M line segments at a resolution

Figure 7: *Hair model rendered with front and back lighting. Notice how light penetrates the hair/mane from the back. The experiment was conducted on AMD Radeon 5800, we listed the performance data at the bottom right corner.*

of 640×640. Our solution is able to achieve a speedup of 2.6 when $K = 16$, i.e., selecting the first 16 fragments. The speedup is more significant when K increase, e.g., we achieve a speedup of 6.2 when $K = 64$.

Our single-pass algorithm, however, is slower than the two-pass algorithm or the OIT technique. Recall that our single-pass solution resolves fragment sorting by maintaining an ordered linked list. It uses more operations for traversing the linked list to locate the insertion point and verifying and changing the successor pointers in global memory. All these operations need to be atomic. In contrast, our two-pass algorithm simply relies on fast $selectandreplace$ operations (Sec.5.1) in register space although it incurs overhead due to context switching. Nevertheless, we envision that the single-pass algorithm may be potentially beneficial in near future when new features such as efficient caching on the GPU memory and faster atomic operations become available.

6.2 Comparison With the Layer-based Solution

The use of K-Buffer allows us to accurately locate the nearest K critical scattering events along a ray for reproducing volume shadows and translucency. Recall that the layer-based techniques such as the Deep Opacity Map (DOM) Occupancy Map(OM) only provide an approximation to the location of the scattering events. Therefore, our rendering results should be more accurate. To demonstrate, we show only the translucency component of the rendered hair. We choose not to include volumetric shadows as shadows appear gray scale and it is more difficult to discern the difference especially when shadow filtering schemes or high resolution shadow maps are used.

The hair fibers are rendered with basic shading and uses a constant alpha value for generating translucency. For fair comparisons, we slice the scene into layers as follows. We first render the hair using a z-min buffer and a z-max buffer to get the accurate depth range. We uniformly partition the space within the range to generate all layers. Note that the original DOM[Yuksel and Keyser 2008] implementation only uses z-min map for space partition. Therefore,

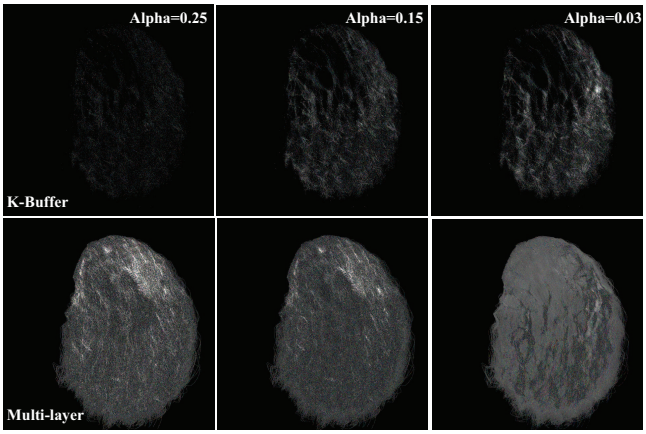


Figure 8: *The difference image (error map) between the ground truth translucency and the rendered result. From left to right, we show results with different hair fiber opacities. Our K-Buffer scheme (top row) produces much more accurate results than the multi-layer based approach.*

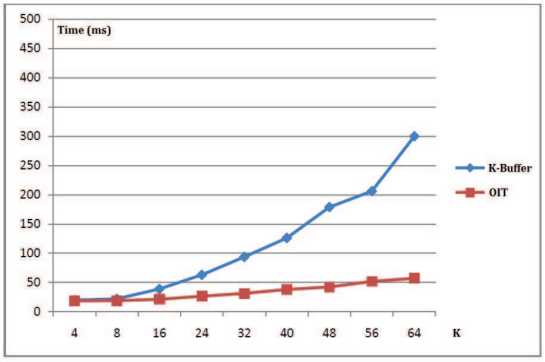


Figure 9: *Performance comparisons between our methods and previous approaches.*

our DOM implementation should be even more accurate.To generate the ground truth, we use the A-Buffer technique to completely order all fragments and then apply ordered alpha blending.

To further illustrate the difference between our and previous technique, we assign each fiber a random color. To compare the results generated by our K-buffer ($K = 16$) and DOM with 32 layers, we compute the difference images between the rendered results and the ground truth and scale the difference images for better illustrations as shown in Fig.8. We use three different alpha values 0.25, 0.15 and 0.03. In our experiments, we find that smaller alpha values will incur more errors for both our and the DOM algorithms. However, ours still outperforms the DOM even with very small alpha. When the alpha value increases (more opaque hair), our K-Buffer results also get closer to the ground truth while DOM remains approximately the same. This is because ours selects the nearest K fragments that contribute most to the final shading while DOM still select elements at evenly sampled range.

6.3 Multi-Scattering Components

Finally, we show the importance of rendering all multi-scattering components for creating realistic looking hair. We have conducted experiments on a number of 3D hair models, some directly acquired and reconstructed from images [Paris et al. 2004; Paris et al. 2008]. Fig.10 shows our rendering results on these models (in different rows). In each row, we show the results with only single scattering (anti-aliased), with volumetric shadows alone, and our results (volumetric shadows, translucency, and anti-aliased).

Results using only single-scattering apparently appear flat, plastic, and overly shiny as they do not model scattering between hair fibers. Adding volumetric shadows significantly improves the realism of hair, by giving useful visual cues of the spatial distributions of hair fibers. Nevertheless, the results still appear stiff and plastic, i.e., they look like faux hair. Our results further integrate translucency with volume shadows and generate soft and semi-transparent looking hair, e.g., on fibers that cover the female model's eye and chin in row 1. The translucency on the mane (row 2) also makes the lion appear more vivid. To further illustrate the our simulation quality, We further cast light from the back of the model and our technique is able to faithfully reconstruct how lights penetrate the hair volume as shown in Fig.7. We refer the reviewers to the supplementary video for more results as well as additional comparisons on results with and without anti-aliasing.

7 Conclusion

We have presented a novel real-time rendering framework for reproducing various multi-scattering effects on complex hair. At the core of approach is a new GPU-based K-Buffer solution to locate fragments (geometric primitives) along a ray. Our K-Buffer solution has a number of advantages compared with the state-of-the-art multi-layer type of approaches. First, our solution can precisely obtain the nearest K fragments along a ray whereas traditional methods only approximate the nearest fragments. Second, our solution works robustly with arbitrarily complex scene and with arbitrary geometry distributions while traditional solutions such as the Deep Opacity Map need to pre-partition the scene into layers and their rendering quality rely heavily on a good partition and uniform geometry distributions. Finally, our approach can simultaneously handle volumetric shadows, translucency, and anti-aliasing and can be implemented on a unified pipeline.

There are a number of future directions that we plan to explore. Our K-Buffer solution is rather generic for handling complex geometry and has the potential to benefit general rendering tasks. For example, we plan to extend our approach to render complex shadowing and scattering effects on other types of objects such as grass and heavily foliaged trees. Our method may also provide an effective solution for rendering multi-reflection and refractions. Finally, as is shown in the result section, our two-pass algorithm currently outperforms the one-pass one, mainly due to the low efficiency in atomic operations. Nevertheless, we hope that our one-pass solution will inspire industry to devise more efficient hardware for conducting atomic operations on the GPU, which would in turn lead to new single-pass alternatives to traditional multi-pass solutions to solve various rendering problems.

References

BAVOIL, L., CALLAHAN, S. P., LEFOHN, A., COMBA, J. A. L. D., AND SILVA, C. T. 2007. Multi-fragment effects on the gpu using the k-buffer. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '07, 97–104.

BERTAILS, F., MÉNIER, C., AND CANI, M.-P. 2005. A practical self-shadowing algorithm for interactive hair animation. In *Proceedings of Graphics Interface 2005*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, GI '05, 71–78.

DUCHENEAUT, N., WEN, M.-H., YEE, N., AND WADLEY, G. 2009. Body and mind: a study of avatar personalization in three virtual worlds. In *Proceedings of the 27th international conference on Human factors in computing systems*, ACM, New York, NY, USA, CHI '09, 1151–1160.

Figure 10: *Results on various hair models. From Left to right, we show the results with only single scattering (anti-aliased), with volumetric shadows alone, and our results (volumetric shadows, translucency, and anti-aliased).*

KIM, T.-Y., AND NEUMANN, U. 2001. Opacity shadow maps. In *In Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, Springer-Verlag, 177–182.

KIM, T.-Y., AND NEUMANN, U. 2002. Interactive multiresolution hair modeling and editing. *ACM Trans. Graph. 21* (July), 620–629.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 385–392.

MARSCHNER, S. R., JENSEN, H. W., CAMMARANO, M., WORLEY, S., AND HANRAHAN, P. 2003. Light scattering from human hair fibers. In *ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, SIGGRAPH '03, 780–791.

MICROSOFT, 2010. Windows directx graphics document.

MOON, J. T., AND MARSCHNER, S. R. 2006. Simulating multiple scattering in hair using a photon mapping approach. In *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, SIGGRAPH '06, 1067–1074.

MOON, J. T., WALTER, B., AND MARSCHNER, S. 2008. Efficient multiple scattering in hair using spherical harmonics. In *ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, SIGGRAPH '08, 31:1–31:7.

PARIS, S., BRICEÑO, H. M., AND SILLION, F. X. 2004. Capture of hair geometry from multiple images. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 712–719.

PARIS, S., CHANG, W., KOZHUSHNYAN, O. I., JAROSZ, W., MATUSIK, W., ZWICKER, M., AND DURAND, F. 2008. Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Trans. Graph. 27* (August), 30:1–30:9.

SALVI, M., VIDIMČE, K., LAURITZEN, A., AND LEFOHN, A. 2010. Adaptive volumetric shadow maps. In *Eurographics Symposium on Rendering*, 1289–1296.

SINTORN, E., AND ASSARSSON, U. 2008. Real-time approximate sorting for self shadowing and transparency in hair rendering. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '08, 157–162.

SINTORN, E., AND ASSARSSON, U. 2009. Hair self shadowing and transparency depth ordering using occupancy maps. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '09, 67–74.

WARD, K., GALOPPO, N., AND LIN, M. 2007. Interactive virtual hair salon. *Presence: Teleoper. Virtual Environ. 16* (June), 237–251.

YANG, J. C., HENSLEY, J., GRUN, H., AND THIBIEROZ, N. 2010. Real-time concurrent linked list construction on the gpu. *Computer Graphics Forum 29*, 4, 1297–1304.

YUKSEL, C., AND KEYSER, J. 2008. Deep opacity maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008) 27*, 2.

YUKSEL, C., SCHAEFER, S., AND KEYSER, J. 2009. Hair meshes. *ACM Trans. Graph. 28* (December), 166:1–166:7.

ZINKE, A., SOBOTTKA, G., AND WEBER, A. 2004. Photo-realistic rendering of blond hair. In *in Vision, Modeling, and Visualization (VMV) 2004*, IOS Press, 191–198.

ZINKE, A., YUKSEL, C., WEBER, A., AND KEYSER, J. 2008. Dual scattering approximation for fast multiple scattering in hair. In *ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, SIGGRAPH '08, 32:1–32:10.