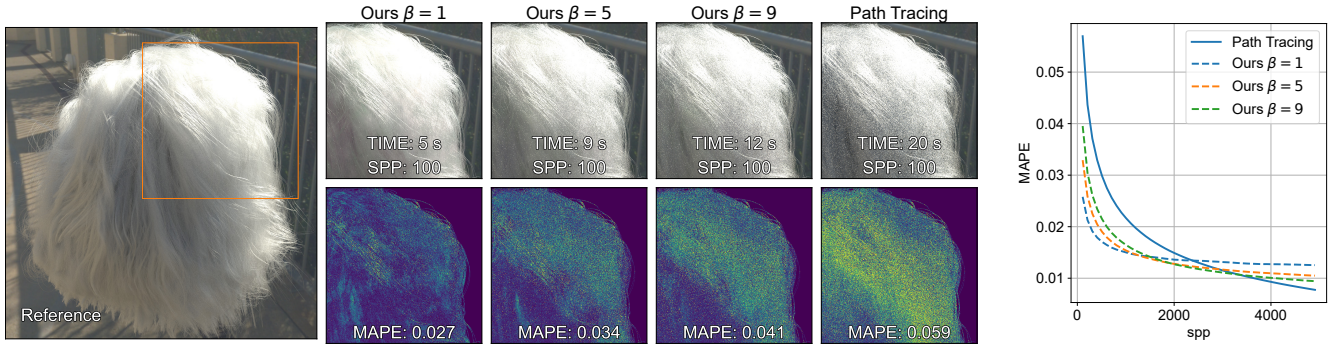
DOI: 10.1111/cgf.14895
Eurographics Symposium on Rendering 2023
T. Ritschel and A. Weidlich
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 4
e14895 *pp. 1 - 13*

Accelerating Hair Rendering by Learning High-Order Scattered Radiance

Aakash KT[† 1,2], Adrian Jarabo[1], Carlos Aliaga[1], Matt Jen-Yuan Chiang[1], Olivier Maury[1], Christophe Hery[1], P. J. Narayanan[2], Giljoo Nam[1]

[1]Meta Reality Labs Research
[2]CVIT, International Institute of Information Technology, Hyderabad (IIIT-H)



(a) (b)

Figure 1: *We propose to use a small multi-layer perceptron (MLP) to approximate higher order scattering in hair without any pretraining, by online learning of the error between biased and unbiased light paths. Our method converges faster compared to path tracing and provides control over it's bias & speedup via the max. path depth parameter* β*. This figure shows the characteristics of our method on messy white hair, with the MAPE metric against a* 10*k spp reference: (a) Our method with* $\beta = 1$ *converges to about one-half the error in one-fourth of the time while tracing the same amount of samples per pixel (100 spp). By increasing* β*, the error and time required to render approaches the time taken by path tracing. (b) Equal spp convergence graphs show that with lower* β *values, our method converges to a lower error faster (since same spp takes less time) but to a biased result at high spp (MAPE of path tracing eventually crosses over to a lower value). With higher* β*, our method's bias reduces eventually approaching path tracing.*

Abstract

Efficiently and accurately rendering hair accounting for multiple scattering is a challenging open problem. Path tracing in hair takes long to converge while other techniques are either too approximate while still being computationally expensive or make assumptions about the scene. We present a technique to infer the higher order scattering in hair in constant time within the path tracing framework, while achieving better computational efficiency. Our method makes no assumptions about the scene and provides control over the renderer's bias & speedup. We achieve this by training a small multilayer perceptron (MLP) to learn the higher-order radiance online, while rendering progresses. We describe how to robustly train this network and thoroughly analyze our resulting renderer's characteristics. We evaluate our method on various hairstyles and lighting conditions. We also compare our method against a recent learning based & a traditional real-time hair rendering method and demonstrate better quantitative & qualitative results. Our method achieves a significant improvement in speed with respect to path tracing, achieving a run-time reduction of $40\% - 70\%$ *while only introducing a small amount of bias.*

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing; Parametric curve and surface models;*** *Volumetric models;*

1. Introduction

Accurately rendering hair & fur is a challenging open problem. The appearance of hair is the result of light-matter interactions at

† Work done during an internship at Meta Reality Labs Research

different scales. On the one hand, local scattering at each individual fiber is responsible for glints and anisotropic highlights. On the other hand, multiple scattering of light between the potentially large number of hair strands generates a colored low-frequency shading, especially in light colored hair.

It is precisely this combination of high-frequency local scattering and potentially large number of global scattering events that make hair rendering computationally expensive: light transport in hair involves tracing rays for a near caustic light path and the final color is obtained only when the radiance from many such paths is averaged. Even with hardware accelerated ray tracing, the large number of scattering events result in slow convergence. Thus, accelerating computation and convergence of path tracing in hair is an active research area.

Several works have been proposed that accelerate multiple scattering by caching the illumination [MM06; MWM08], approximating it using diffusion [YSJR17], or by means of asymptotic approximations under certain assumptions such as *dual scattering* [ZYWK08]. These methods are either too complex while gaining little computational advantage, or make assumptions about the scene & lighting or impose strong approximations lacking the visual fidelity of hair.

Our work starts from the observation that light transport in hair presents two different regimes: (1) A low order scattering term responsible for highlights, glints and a directional glow, (2) A high order scattering term which further contributes to this glow along with a diffuse-like scattering component. These regimes are not exclusive to hair and manifest in other discrete media as well. We thus follow a similar approach to methods proposed for granular media [MPH*15; MPG*16] and decompose the light transport of hair into these regimes.

Similar to these methods for granular media, our aim is to avoid precomputations and assumptions on the scene structure. However, unlike such methods and other hair rendering techniques, we aim for: (1) constant time to render the high-order scattering component, and (2) controllable bias and speedup of the renderer with an accurate estimate of render time. To this end, we propose to use a small multi-layer perceptron (MLP) to approximate the high order scattering component on-the-fly.

Specifically, in our approach, rendering a frame consists of three stages: In the first stage, we train an MLP, on a small fixed-size dynamic set of light path pairs within the hair volume. Each pair consists of a *short path* which can be chosen according to a given render time budget, and a *long path* which is an extension of the short path until termination. The MLP is tasked to learn the error between these paths. In the second stage, we trace one short path per pixel from the camera and accumulate its radiance. In the final stage, the error of these short paths is inferred from the MLP and corrected for, resulting in a final radiance accounting for low and high order scattering. This process is repeated every frame which progressively renders the scene. Since the MLP is always trained and inferred on fixed size data, the time required for the first & third stages is constant. This coupled with the fact that short paths can be adjusted for a specific time budget gives an upper bound on the render time.

We derive the quantity learnt by the MLP and show that it is bounded and converges. We then describe a methodology to efficiently train and use the MLP within the rendering loop. We demonstrate that our method renders the true color of the hair due to multiple scattering in a short time as opposed to path tracing which gradually recovers it. We further show that in the best case, our method achieves a $40\% - 70\%$ speedup in run-time at the cost of minimal bias. We analyse our method extensively and show that it naturally provides a control over its bias and speedup by controlling the maximum path depth β of these short paths. In effect, our method can trade off less bias for more speedup & vice versa.

Fig. 1 demonstrates two characteristics of our method on messy white hair: (a) smaller error & time required to trace equal number of samples per pixel as path tracing, (b) controllable bias & speedup.

2. Related Work

Physically-based hair scattering. The seminal work of Marschner and colleagues [MJC*03] modeled the scattering of hair as a bidirectional curve scattering function (BCSDF) [ZW07], approximating each fiber as a dielectric filament with circular cross section. Inspired by Marschner's model, several works have been proposed improving its accuracy [dFH*11; HHH22; XWM*20], generalizing to elliptical cross sections [KM17; BP21], or making it more practical for production scenarios [PHVL15; SPJT10; CBTB16]. While advanced hair scattering models can improve the efficiency of rendering, our work is independent of specific a scattering model used and provides a general rendering acceleration solution, with a focus on accelerating computations of multiple scattering between hair strands.

Accelerating multiple scattering. Multiple scattering is responsible of hair coloration, specially in light colored hair, but extremely expensive due to the long light paths and forward scattering. Moon and colleagues proposed to use photon mapping to cache the radiance at the hair volume [MM06], which was later improved by encoding radiance in spherical harmonics [MWM08]. Another approach for accelerating multiple scattering is to separate light transport into different regimes (ballistic, directionally diffuse, diffuse): Dual scattering [ZYWK08] approximated multiple scattering by deriving closed-form approximations of the near and far scattering components. Meng et al. [MPH*15] and Müller et al. [MPG*16] used a similar light regime decomposition in the context of particulate media. Dual scattering however fails to reproduce the soft look and saturation visible in light colored hair. Yan et al. [YSJR17] extended dual scattering to fur, by adding a diffusion term. For human hair, their method falls back to dual scattering.

Hery and colleagues [FHP*18, Chap.7] proposed to reduce the multiple scattering albedo of hair (i.e. its overall color) by some factor and compensate for the energy loss by multiplying the illumination by the same factor. This effectively reduced the path weight, increasing the possibility of early termination which led to increased efficiency. They demonstrated that by using a factor of two, render times decreased by 45% while achieving a similar look. However, increasing the factor will result in shorter paths with larger deviation from the desired look.

Zhu et al. [ZZW*22] accelerated multiple scattering in fur by reducing the fur density, and increasing the fiber thickness using learned aggregate scattering behaviour; this approach is similar to the similarity theory [WPW89; ZRB14] where the optical parameters of general media are altered for reducing the density of the medium and therefore reducing the amount of scattering events. However, their approach is not directly applicable to hair with long unordered strands, unlike fur. This makes the aggregation of hair difficult.

Our method takes a different approach, avoiding precomputations and accelerating hair rendering by inferring high-order scattered radiance in hair using an online-trained & small multi-layer perceptron.

Deep Learning in Rendering. A number of works have focused on accelerating global illumination using neural networks. As mentioned above, Zhu et al. [ZZW*22] aggregate hair strands and learn the aggregated scattering with an MLP. In the context of optically thin, high albedo volumes, Kallweit et al. [KMM*17] render atmospheric clouds by training a network to learn the spatial and directional distribution of radiant flux from cloud exemplars. For translucent, optically thick materials, Vicini et al. [VKJ19] proposed to learn the Bidirectional Subsurface Scattering Distribution Function (BSSRDF) of a target object with any shape and material properties. Also for translucent objects, Che et al. [CLZ*20] infer material properties from images by training an auto-encoder with a differentiable Monte Carlo volume renderer as a decoder. All of these methods pre-train neural networks for lower-dimensional sub-problems (the scattering functions). Our approach on the other hand, works directly in the high-dimensional path space, and learns a mathematically well-defined function of the total radiance at a given ray.

With a more generic goal, Müller et al. [MMR*19] presented neural importance sampling, an approach for online learning of the directional distributions and path sampling and guiding for simulation of light transport, as well as a neural version of control variates [MRKN20]. Closer to our work, Müller and colleagues [MRNK21] presented a neural radiance caching technique that approximates a 5D radiance field of a scene. Differently, we tackle the specific case of hair volumes, which exhibit high frequencies both spatially (i.e. there are around 200k hair strands in a human head) and angularly (highly directional, anisotropic scattering). Such properties, which are particularly enhanced for light colored hair where light paths undergo many bounces inside the volume, make the previous techniques either impractical or lack the visual fidelity. Instead of learning the complex 5D radiance field in the hair volume, we train a similar small MLP to learn and infer the error between biased low-order scattered radiance and full unbiased scattered radiance (effectively, high-order scattered radiance).

3. Preliminaries

We begin with a recap on path tracing, which also helps establish notation of our paper. We also briefly describe Russian roulette, a technique to achieve early termination of low energy paths. Our method is described in Sect. 4.

3.1. Path Tracing

The radiance L at arriving at a pixel is modeled by the path integral [Vea98] as:

$$L = \sum_{k=1}^{\infty} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{1}$$

where Ω_k is the space of light paths $\bar{\mathbf{x}} = \mathbf{x}_0..\mathbf{x}_k \in \Omega_k$ of length k with $k+1$ path vertices. $\mathbf{x}_k$ and $\mathbf{x}_0$ are 3D points placed on a light source and the sensor respectively, and the differential measure $\mathrm{d}\mu(\bar{\mathbf{x}})$ models the area/volume integration for each vertex in the path. The path contribution $f(\bar{\mathbf{x}})$ is defined as:

$$f(\bar{\mathbf{x}}) = L_e(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1})\, T(\bar{\mathbf{x}}), \tag{2}$$

where L_e is the emitted radiance of the light source at $\mathbf{x}_k$ towards $\mathbf{x}_{k-1}$, and $T(\bar{\mathbf{x}})$ is the path throughput:

$$T(\bar{\mathbf{x}}) = \prod_{i=1}^{k-1} S(\mathbf{x}_{i+1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i-1}) G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i), \tag{3}$$

where S is the scattering kernel (e.g. the BCSDF in the case of hair) and G the the geometric term between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$.

Path tracing numerically approximates Eq. (1) using the unbiased Monte Carlo (MC) estimator:

$$L \approx \langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{\mathbf{x}}^i)}{p(\bar{\mathbf{x}}^i)}, \tag{4}$$

where $\bar{\mathbf{x}}^i$ is a randomly generated light path sampled from $\Omega \in \{\Omega_1, \Omega_2, ..., \Omega_\infty\}$ with probability $p(\bar{\mathbf{x}}^i)$. In practice, path tracing recursively builds this path by sampling new vertices starting from $\mathbf{x}_0$, until a light source is reached. In case of scenes with hair and especially light coloured hair, the radiance estimate $\langle L \rangle$ is dominated by multiple scattering, which means most of the sampled paths $\bar{\mathbf{x}}^i$ need to be long and should also cover a wide range of path lengths. However the energy contribution of deeper path vertices drops quickly for a few of these lights paths, thus lowering the overall efficiency.

3.2. Path Termination with Russian Roulette

Russian Roulette (RR) is used to improve efficiency of $\langle L \rangle$ by probabilistically terminating long paths with low energy. During recursive sampling of path vertices, at each vertex or scattering event $\mathbf{x}_j$ on $\bar{\mathbf{x}}$, the path is terminated by a probability $p_{\mathrm{RR}}(\mathbf{x}_j)$. The integral at the terminated vertex is not evaluated and is zeroed. We note that an estimate that better represents this integral can be used instead [SSK03] which may help to further reduce variance, and our method as is can potentially provide this estimate. There are multiple ways for computing $p_{\mathrm{RR}}(\mathbf{x}_j)$, based on the single scattering albedo at $\mathbf{x}_j$ [AK90], the expected incident radiance [VK16; RGH*22], or the accumulated subpath weight $w(\bar{\mathbf{x}}_j) = T(\bar{\mathbf{x}}_j)/p(\bar{\mathbf{x}}_j)$ [PJH16], with $\bar{\mathbf{x}}_j = \mathbf{x}_0..\mathbf{x}_j$.

We use RR based on accumulated subpath weight. Typically, RR termination is applied only after the first n path vertices (for example, PBRT [PJH16] uses $n = 3$). However, we apply RR from $n = 1$ for our method and path tracing. We note that our method and formulation is independent of the choice of n.

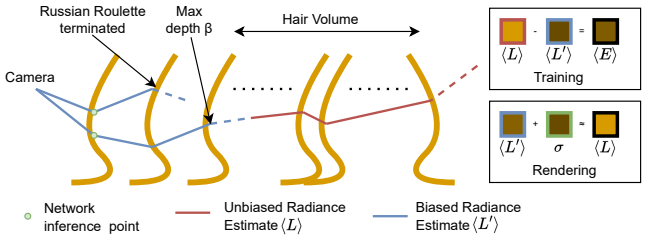


Figure 2: *We learn the error $\langle E \rangle$ (Eq. 6) between radiance estimates of long unbiased paths $\langle L \rangle$ (red, Eq. 1) and short biased paths $\langle L' \rangle$ (blue, Eq. 5) using an MLP. The short paths are traced until termination by Russian Roulette (RR) or upto a small max depth β and the long paths are traced until termination by RR. The network σ is trained to reproduce $\langle E \rangle = \langle L \rangle - \langle L' \rangle$. During rendering, the radiance at the camera is computed as the radiance of the short path plus the error inferred by the network at the primary path vertex (green). The thick yellow lines depict hair strands.*

4. Method

In this section, we describe our method to approximate higher order radiance in hair using an MLP. We begin by setting a maximum depth $\beta << \infty$ in Eq. (1), giving the radiance L' arriving at the pixel:

$$L' = \sum_{k=1}^{\beta} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}). \tag{5}$$

We estimate $\langle L' \rangle$ inplace of $\langle L \rangle$ during rendering. Modifying Eq. (1) in this way effectively sets a maximum depth for path termination, which introduces bias/error in $\langle L' \rangle$, given by:

$$\begin{aligned} E &= L - L' \\ &= \sum_{k=1}^{\infty} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) - \sum_{k=1}^{\beta} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) \\ &= \sum_{k=\beta+1}^{\infty} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \end{aligned} \tag{6}$$

which we can estimate using the Monte Carlo estimate $\langle E \rangle$ in the same fashion as Eq. (4). This error estimate $\langle E \rangle$ is bounded and converges in expectation, and captures the missing higher-order scattering component in $\langle L \rangle$. In effect, $\langle E \rangle$ represents the error between the expected value of unbiased path contributions & the contribution from early terminated paths. Furthermore, $\langle E \rangle$ is itself unbiased, since its form is similar to Eq. (4), except that the paths $\bar{\mathbf{x}}$ are instead sampled from $\Omega \in \{\Omega_{\beta+1}, \Omega_{\beta+2}, ..., \Omega_{\infty}\}$.

We task an MLP σ to learn $\langle E \rangle$ at the primary path vertex $\mathbf{x}_1$, given the view direction $\omega = \frac{\mathbf{x}_0 - \mathbf{x}_1}{||\mathbf{x}_0 - \mathbf{x}_1||}$ and the hair tangent $\mathbf{t}_1$ at $\mathbf{x}_1$:

$$\sigma(\mathbf{x}_1, \omega, \mathbf{t}_1) \approx \langle E \rangle. \tag{7}$$

Fig. 2 shows an overview of our method, which computes $\langle E \rangle$ to train the network σ and uses the network's inferred output at the primary path vertex to correct for the bias in $\langle L' \rangle$.

Discussion. We aim to avoid precomputations, which precludes training the MLP σ offline. Therefore, limited by the time constraints of online training and inference, we use a small MLP. Furthermore, the computation of $\langle E \rangle$ to train the network is done on a small percentage ($\sim 1\%$) of the total paths traced per frame. In Sec. 5, we describe our implementation to efficiently train σ during rendering.

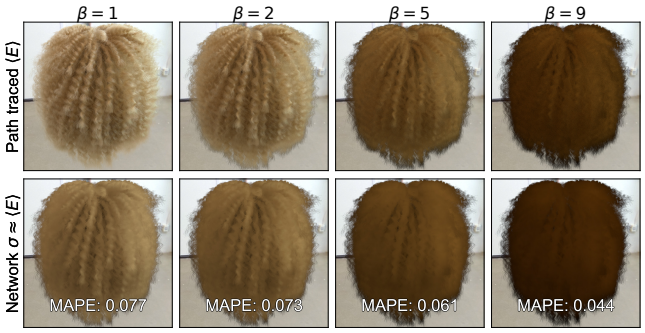


Figure 3: *Visualization of the target function $\langle E \rangle$ and its learned approximation for $\beta = 1, 2, 5$ and 9. For lower values of β, the target function has higher frequencies which the network is unable to reproduce. At higher β's, the target function is lower frequency and the network can better represent it, which is also confirmed by the MAPE values. The full rendering can be found in Fig. 4*

Learning an estimate of the error $\langle E \rangle$ instead of directly learning an estimate of the full radiance field has two major advantages:

- The frequency of $\langle E \rangle$ over paths $\bar{\mathbf{x}}$ starting at $\mathbf{x}_0$ and with a shared common first hair vertex $\mathbf{x}_1$ is low: The magnitude of L' closely follows the magnitude of L for their single sample MC estimates thanks to RR termination, given that these estimates share the same path vertices. This makes $\langle E \rangle$ smooth over paths and of lower magnitude.
- The estimate $\langle E \rangle$ is a high-order radiance estimate, and thus has lower spatial frequency compared to $\langle L \rangle$, which has high frequencies as a result of complex primary visibility within hair and high dynamic range BCSDF lobes. This also follows from the observation that high order scattering in hair is diffuse-like [MJC*03; dFH*11; MM06].

We demonstrate this by visualizing the target signal $\langle E \rangle$ and its learned approximation with σ in Fig. 3, for $\beta = 1, 2, 5$ & 9. For lower β's, $\langle E \rangle$ has higher frequencies (highlights) due to the specular & directional nature of the hair BCSDF. Although the network is able to sufficiently capture spatial variation, it does not reproduce these highlights. For higher β's, the frequency is attenuated and the network is better able to reproduce the target signal. This is also confirmed by the mean absolute percentage error (MAPE) [LKB*22] values computed with respect to $\langle E \rangle$, which are lower for higher β's. Thus, choosing $\langle E \rangle$ as the target for the MLP makes it robust and better represent the signal, more so since we are limited to a small MLP with limited learning capacity.

Our second goal of controllable bias and efficiency is also naturally achieved by means of the parameter β. For large β, $\langle E \rangle$ has even lower frequency and amplitude (since the error vanishes) and thus the network's output closely matches the ground truth. In the limit for very large β, $\langle E \rangle$ approaches zero, and our method is equivalent to unbiased path tracing. This comes at the cost of efficiency, since the rendering cost with $\langle L' \rangle$ approaches that of $\langle L \rangle$,

both consisting of long paths. On the other hand, for a very small β, $\langle E \rangle$ increases in frequency, starting to reach the network's learning capacity. This not only results in an overall increase in bias, but the efficiency also increases since rendering with $\langle L' \rangle$ now requires shorter paths to be traced.

In summary, learning the *error* instead of the full radiance field makes the network learning robust and better match the target signal. Furthermore, our formulation for learning this error naturally allows for control over the bias and speedup. In the next section, we describe our implementation for efficiently training σ while rendering & give details on the network structure.

5. Implementation

We implement our approach using CUDA and OptiX [PBD*10] for hardware accelerated ray tracing and use tiny-cuda-nn [Mül21] for efficient network evaluation and training. We use path tracing as a baseline within the same framework. For better convergence, we use Quasi Monte Carlo with Sobol random number generation for our method as well as path tracing. Furthermore, all direct lighting computations use multiple importance sampling [Vea98] with BCSDF and environment map importance sampling. Alg. 1 shows the pseudo code for rendering a frame using the inferred error from the network, while also training it per frame.

Training. The training procedure is given in lines 1-10. We train the network on paths originating from randomly sampled points that are visible to the camera. Given a sampled point x on hair, we trace two paths: A short path with maximum depth β, and a long path which extends this short path until termination (lines 2-6). This is depicted by the use of the same random number sequence for tracing these paths (line 4). In practice, the computations of the short path are reused. Note that both short and long paths are terminated by RR, although short paths are strictly terminated at bounce β. The sampled short and long paths are used to estimate Eq. (5) and Eq. (1) respectively. We then compute the error (Eq. (6)) in line 7, and update the network weights using the relative L_2 luminance loss [LMH*18; MRNK21] (lines 8-10). We use the same training setup (optimizer, learning rate) as [MRNK21].

Rendering. Rendering a frame consists of two stages, which are described in lines 17-28 in Alg. 1. In the first stage, the network is trained for one second (lines 19-21), ensuring that large errors from an untrained network are not propagated into the final render. A crucial point to note is that this training is *frame independent*, which allows for more initial training iterations and a better converged network. The next stage is responsible for both training the network and rendering the frame. After the initial training and before a sample is traced, the network is trained on a fixed size (size N), small & dynamically sampled set of path pairs (lines 23-25), starting from a visible point. Using a small set is crucial since training is repeated for every sample that is traced. In our implementation, we set the number of training samples $N = 16,384$, which is 1% of our render resolution of 1024×1024. After training, for each pixel we trace short paths with maximum depth β and estimate Eq. (5) (line 26, 27, 13). We also record a G-buffer at the primary intersection of the camera rays. We then evaluate the network to get inferred error estimate E' and add it back to the biased short path estimate L' (lines 14-16).

ALGORITHM 1: Rendering a frame while training the MLP

```
1  Def trainNetwork(β, x):
2      t = getHairTangent(x)
3      ω_o = norm(cam.origin - x)
4      ξ = randomSequence()                    // Seq. of rand. num
5      L = pathTrace(x, ω_o, k ∈ {1,∞}, ξ)     // Eq. (1)
6      L' = pathTrace(x, ω_o, k ∈ {1,β}, ξ)    // Eq. (5)
7      E = L - L'                              // Eq. (6)
8      E' = σ[hashGrid(x), ob(ω_o), ob(t)]     // MLP σ forward
9      loss = L_2^rel(E, E')                   // Rel. L2 luminance
10     loss.backward()                         // Update weights of σ
11 Def renderPixel(pix, β):
12     /* Trace short path & get G-Buffer      */
13     L', G_buf = pathTraceCam(pix, k ∈ {1,β})  // Eq. (5)
       /* Evaluate MLP σ on G-Buffer           */
14     E' = σ[hashGrid(G_buf.x), ob(G_buf.ω_o), ob(G_buf.t)]
       /* Add inferred error                   */
15     L = L' + E'
16     return L
17 Def renderFrame(β: Max. depth, SPP: spp, N: # Train Samples):
18     /* Init. training for 1 sec             */
19     while time ≤ 1sec do
20         x = sampleVisible()
21         trainNetwork(β, x)
       /* Render Loop                          */
22     for sample in SPP do
23         for n in N do
24             x = sampleVisible()
25             trainNetwork(β, x)
26         for pix in PIXELS do
27             L = renderPixel(pix, β)
28             accumulate(pix, L)
```

Network structure. We use a 64-neurons wide and 2-layers deep MLP σ with ReLU activations (except the last layer). We use a multi-resolution hash grid (hashGrid)[MESK22] for encoding the 3D point x, and one-blob (ob) encoding [MMR*19] for the view direction ω_o and the hair tangent t (Alg. 1, line 14). Multi-resolution hash grids can better represent the target signal than frequency encoding with a minimal performance penalty, and have also been shown to work well for radiance caching [MESK22].

Discussion. Since we use OptiX and CUDA in our implementation, the loops in line 19, 23, 26 of Alg. 1 are parallelized across GPU cores. Thus, the function *sampleVisible* returns a buffer of visible points on hair. Consequently, the function *trainNetwork* operates on this buffer in one single pass. Similarly, the function *renderPixel* operates on a buffer of all image pixels.

6. Results, Analysis & Comparison

In this section, we show the rendering results of our method and analyse its characteristics. We also compare against Neural Radiance Caching [MRNK21], which uses a similar MLP for inferring the full radiance field, and Dual Scattering [ZYWK08] which is a real-time method for rendering multiple scattering in hair. We

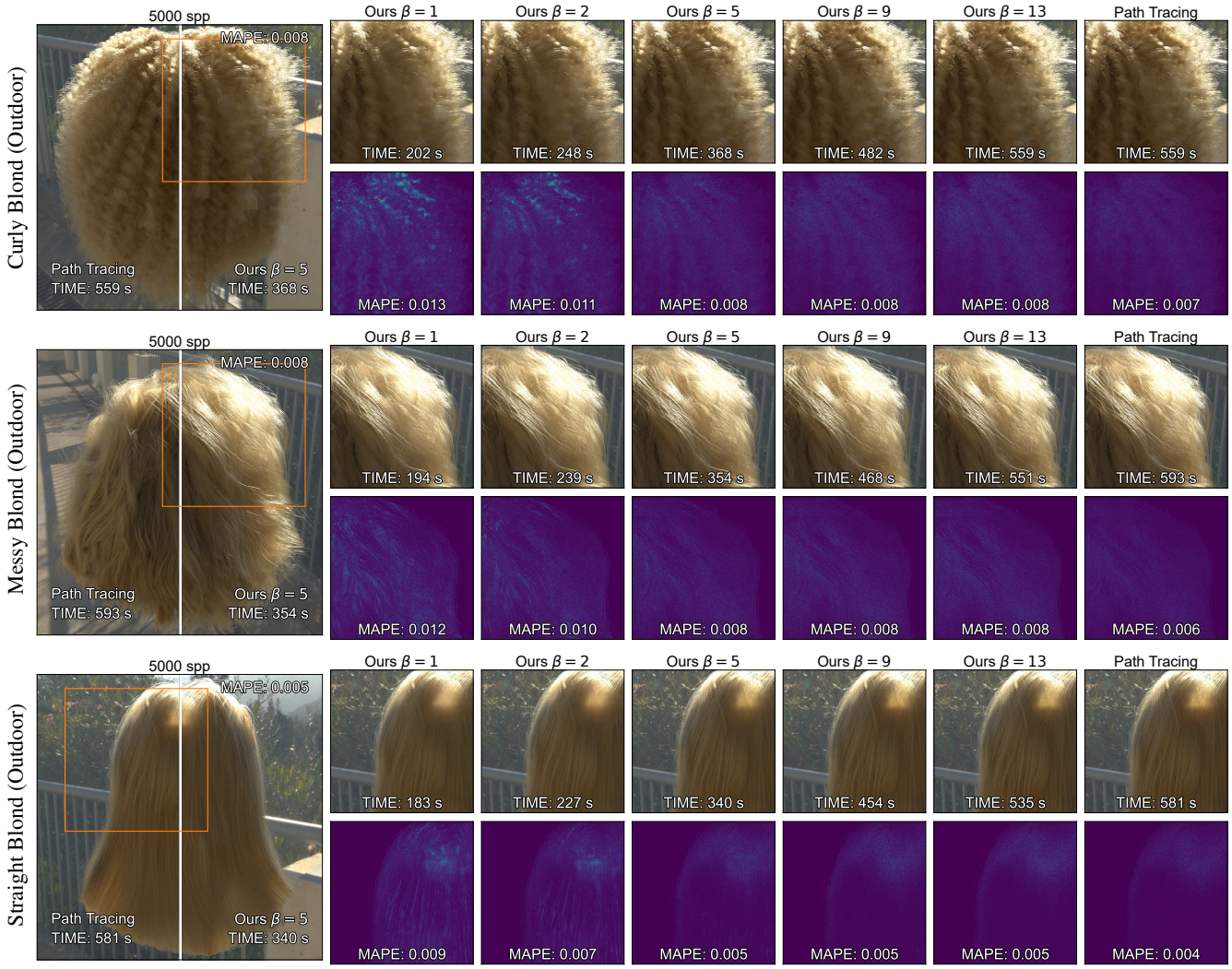


Figure 4: *We show results of our method for* $\beta = 1, 2, 5, 9$ *&* 13 *and compare to path tracing at 5k spp, along with false colour difference images with respect to path traced reference (10k spp). We also report time (in seconds) and the MAP error for all variants. Our method converges in error to unbiased path tracing for large* β*, while exhibiting larger bias at lower values. On the other hand, at lower* β*, the time taken for 5k spp is halved to that of path tracing, increasing for larger values. This demonstrates the control that our method provides over it's bias & speedup.*

demonstrate better qualitative & quantitative results than both these approaches.

To evaluate our method on a spectrum of challenging scenes, we use three different hair styles: Curly, Messy & Straight, and two different HDR lighting setups: Indoor & Outdoor. Table 1 shows the statistics of different hair styles. Rendering is done with Disney's hair BCSDF model [CBTB16] for four absorption coefficients resulting in different hair colors: White $\sigma_a = (0.01, 0.01, 0.01)$, Blond $\sigma_a = (0.06, 0.1, 0.2)$, Brown $\sigma_a = (0.2, 0.3, 0.5)$ and Black $\sigma_a = (3.35, 5.58, 10.96)$. We note that significant multiple scattering occurs with small σ_a values and our goal is to accelerate its computation. Thus, we majorly show results on Blond & White hair. Results on other σ_a values are shown for completeness. We also note that renderings of our method are generated by initially training the network for the one second (Sect. 5). For a fair comparison, we start path tracing and all other methods without the initial one second gap (i.e. start rendering from 0 seconds), unless otherwise stated. Furthermore, the network continues training for each sample that is traced, as mentioned in Sect. 5.

6.1. Rendering Results

High sample count, equal spp. Fig. 4 shows our results at various values of β at 5k samples per pixel (spp) compared to path tracing for the same sample count. We also show false color difference images and report the MAPE metric with respect to a path traced reference rendered at 10k spp. We also show render times

Figure 5: *Equal spp (100 spp) renders of our method for* $\beta = 1, 2, 5, 9, 13$ *compared to path tracing. For white hair, our method with* $\beta = 1$ *achieves one-third the error in one-fourth the time. These metrics approach path tracing with increasing* β*. For darker hair, the gain is less pronounced but still significant. Our method behaves equivalently to path tracing on black hair.*

(in seconds) for both methods. This comparison illustrates the behaviour of the bias introduced by our method for a high spp and also demonstrates the bias-speedup control. At $\beta = 1$, our method achieves the maximum speedup of $\sim 60\%$ with respect to path tracing. However, the MAP error is also larger due to higher bias from the network, as also shown by the difference images. In general,

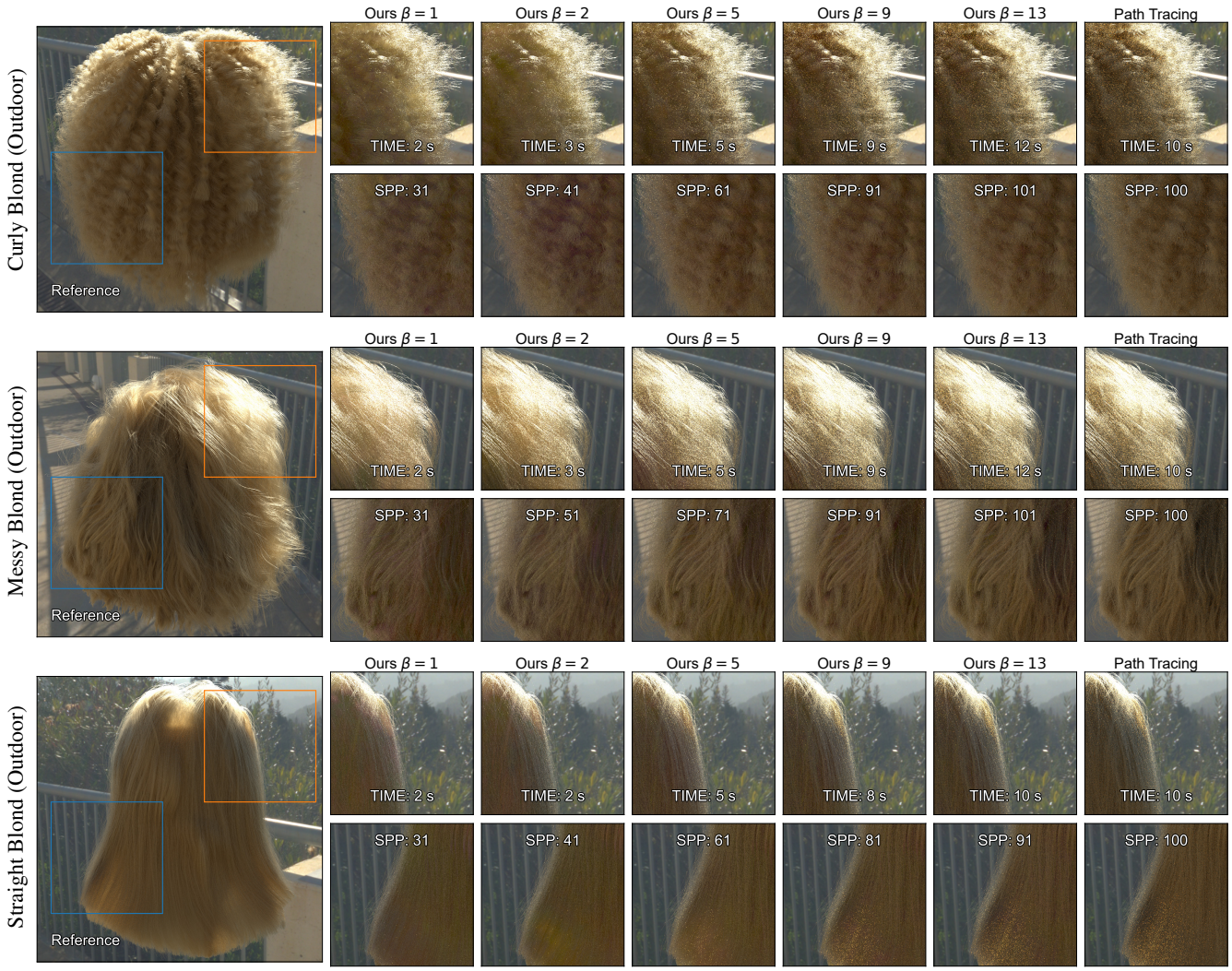


Figure 6: *We calculate variance of path tracing at 100 spp, and show results, timings & spp required for our method with* $\beta = 1, 2, 5, 9$ *&* 13 *to reach the same variance. The variance of each method is computed as the mean-squared error (MSE) with it's own fully converged counterpart. This figure in combination with Fig. 4, illustrates the effect of* β *to trade bias in favor of performance, and shows that our method achieves better performance than path tracing for mid-range* β *values.*

for low values of β, the error is larger at deeper points in the hair volume or where there is significant multiple scattering. This error is largely due to the network reaching its learning capacity and averaging its output. At higher values of β, our method produces results that closely match path tracing, with similar run-time. In the end, after a certain threshold of β (e.g. 13), our method gains little benefit and could lead to increased run-time from the additional overhead of network training and inference steps.

Low sample count, equal spp. We show equal spp (100 spp) renderings of our method on different scenes in Fig. 5. This figures serves to illustrate the benefit of our method at low spp. For white hair which has the most multiple scattering, our method with $\beta = 1$ achieves about one-third the error in about one-fourth the time. These values approach that of path tracing with increasing β. With blond hair and in general darker hair, the gain in run-time and error decreases, albeit still being significant. On black hair our method matches the performance and quality of path tracing for all β values. Visually, our method has lesser noise and takes lesser

Table 1: *Number of strands of different hair styles, along with run-times & MAPE of our method with* $\beta = 1$ *and Path Tracing (PT) for 100 spp. Metrics are calculated with blond hair* $\sigma_a = (0.06, 0.1, 0.2)$ *and indoor HDR lighting. Our method converges to around half the error in half the time compared to path tracing.*

		Blond (Indoor), 100 spp			
		Ours ($\beta = 1$)		PT	
Hair Geom.	# Strands	Sec.	MAPE	Sec.	MAPE
Curly	60k	**4**	**0.028**	10	0.064
Messy	100k	**4**	**0.026**	10	0.056
Straight	100k	**4**	**0.020**	10	0.046

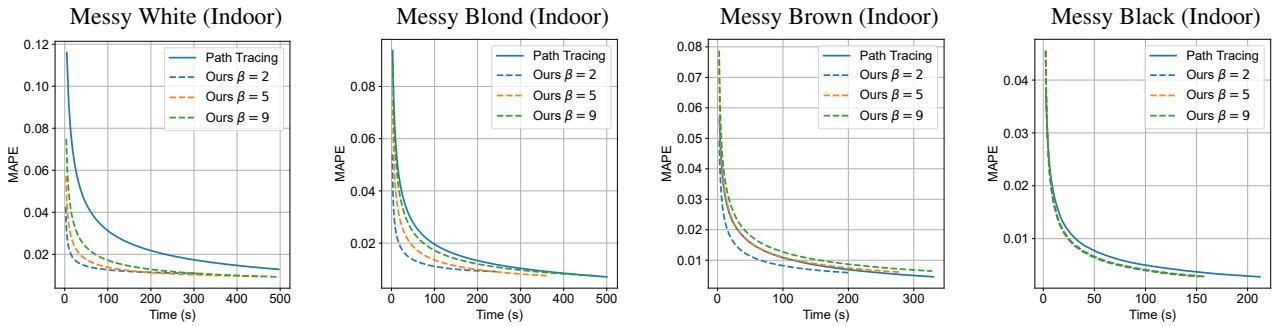


Figure 7: *Equal time convergence graphs with the MAP error metric. Our method showcases the most benefit for light hair, where the convergence is significantly faster than path tracing for lower* β *values. For higher values, the convergence starts to approach path tracing. For darker hair, the convergence is closer to that of path tracing, for all* β*'s. We refer the reader to the supplementary video to better understand the convergence behaviour.*

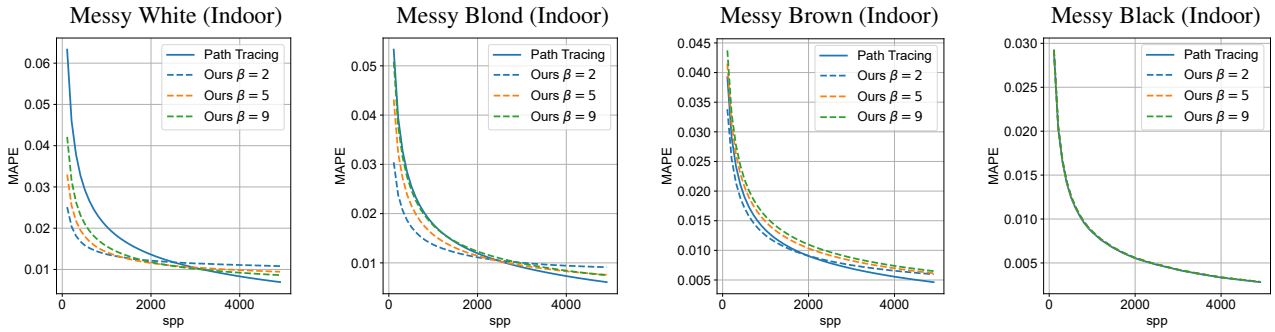


Figure 8: *Equal spp convergence graphs with the MAP error metric. Our method converges to a lower error for the same spp, especially for light hair. These graphs also show that at high spp, our method converges to a biased result, and this bias can be reduced by using higher* β *values.*

time to trace 100 spp compared to path tracing, especially for light coloured hair. Table 1 consolidates the run-time and MAPE values for blond hair with different hair styles rendered for 100 spp with indoor HDR lighting.

Equal variance. We analyze the convergence characteristics of our method by comparing equal levels of variance for different β against path tracing (Fig. 6). For each β, the variance is computed as the render's mean-squared error (MSE) with respect to it's own fully converged counterpart (10k spp). This figure in combination with Fig. 4, illustrates the effect of β to trade bias in favor of performance. For lower β, our method can achieve the same levels of variance as the path tracing with much fewer spp and lower time. This is at the cost of progressively introducing more bias as β decreases, that nevertheless remains reasonable at its peak $\beta = 1$, depending on the target application. Our method achieves better performance for both low and high β until up to very high values, where obviously the learning overhead does not compensate for the very little energy left, in which case the performance of path tracing is better.

Convergence. To further analyze the convergence characteristics, we plot the MAP error against time for our method and path tracing in Fig. 7. We also similarly plot the MAP error against spp in Fig. 8. These graphs are plotted for messy white, blond, brown & black hair styles for $\beta = 2, 5, 9$. Note that, as mentioned before, we train our network for one second before rendering starts, while the rendering is started immediately for path tracing. These plots show that in general, our method converges to a lower error faster, especially for light hair, even with the additional training delay at the beginning. Note however that for larger values of β, our method's convergence approaches that of path tracing, as at these values the paths are longer and there is less bias from the network. Since path tracing converges to an unbiased solution, unlike our method, there always exists a time at which the error of path tracing will crossover to a lower value than that of our method. This is shown in Fig. 8, where for very large spp, the error of path tracing is lower. For darker hair, both extremes of β result in similar convergence closely following that of path tracing. To that end, our method is less beneficial for darker hair. We refer the reader to the supplemental video for a better judgement of the convergence behaviour for different hair styles and β's.

Efficiency. To further study our method's characteristics with respect to β, we analyze it's efficiency for 5k spp renders with respect to a 10k spp reference. We compute efficiency as $\frac{1}{\text{time}\cdot\text{error}}$, with time in seconds and the MAP error. Note that efficiency is typically computed using *variance* instead of *error*; however, we use the lat-

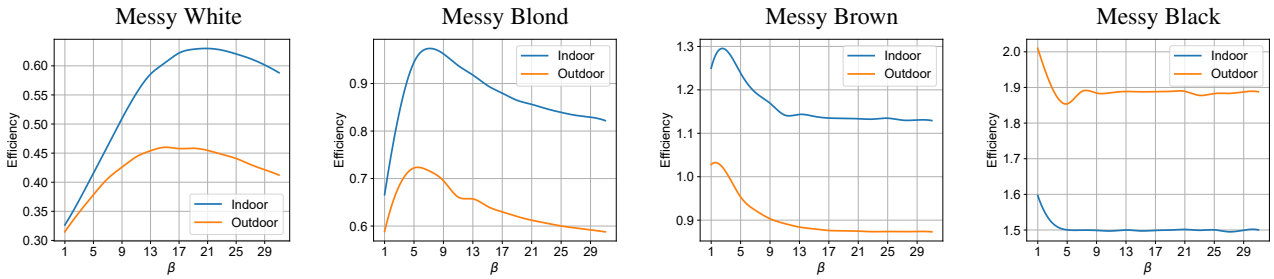


Figure 9: *We plot* β *vs. efficiency graph to analyse the characteristics of our method. Efficiency is computed as* $\frac{1}{time \cdot error}$ *with time in seconds & the MAP error. For white hair, maximum efficiency is achieved at larger* $\beta \in \{13, 25\}$*. This peak shifts as the hair gets darker. Within a hair style, the maximum efficiency depends on the lighting condition, however the peak is roughly in a similar* β *range.*

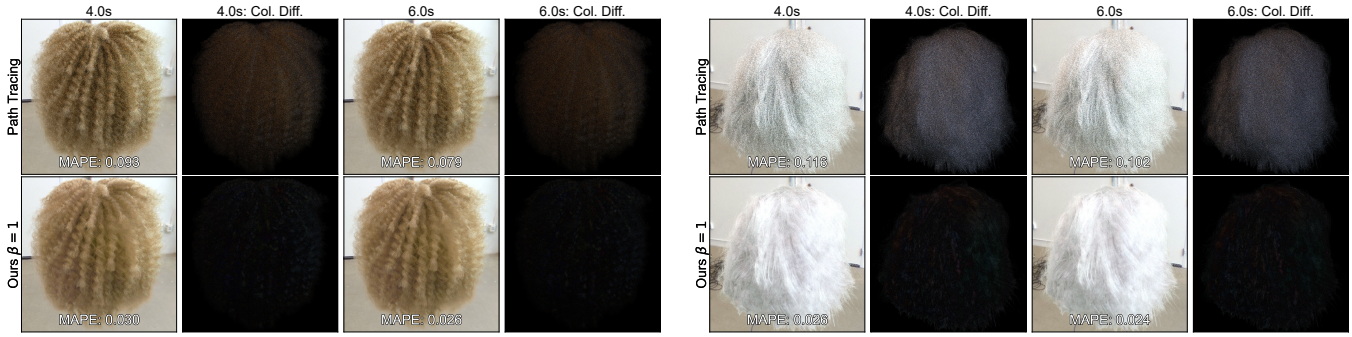


Figure 10: *Our method is beneficial at low spp & small render times to get an accurate estimate of the true color of hair, as shown in the color difference images which have a larger color component for path tracing.*

Table 2: *Decomposition of timings of our method and path tracing (PT). For our method, we show timings for tracing rays upto a max path depth* β *along with training and inference timings. All timings are in milliseconds (ms) and averaged over ten runs.*

	Ours time (ms) for 1spp					
Hair Geom.	$\beta = 2$	$\beta = 5$	$\beta = 9$	Train	Infer	PT
Curly	46	69	96	3	0.05	112
Messy	44	69	89	2	0.06	117
Straight	45	64	97	2	0.06	116

ter to account for both bias and variance, which helps us evaluating which value of β achieves the maximum speedup with the least bias. The plots are shown in Fig. 9 for messy white, blond, brown & black hair styles, each with two different lighting setups (indoor & outdoor). For white hair, maximum efficiency is achieved at larger $\beta \in \{13, 25\}$. This peak shifts as the hair gets darker, ultimately being the most efficient at $\beta = 1$ for black hair. Within a hair style, the maximum efficiency depends on the lighting condition (max. value is different for indoor & outdoor lighting). However, the peak is roughly in the same β range for both lighting types.

Progressive, equal time. We demonstrate another benefit of our method at low spp & small render times: to get an accurate visual estimate of the true color of hair. Fig. 10 shows equal-time renders of our method with $\beta = 1$ compared to path tracing, sampled at different time budgets, for curly blond and messy white hair. We also show a color difference image with respect to a 10k spp path traced reference. Path tracing gradually recovers the color with time as more samples are traced. This can be seen in the difference images where they have a larger color component for path tracing. On the other hand, the difference images of our method show that the difference is majorly in the intensity. Indeed, as compared to path tracing, our method produces the multi-scattered color the hair from the start. This is useful in situations where the parameters of the hair need to be constantly adjusted to achieve a desired look. In such situations, it is beneficial to have an accurate estimate of how the hair looks from the start, allowing quicker iteration.

Upper bound on run-time. By fixing β, we get an estimate of the maximum number of rays that need to be traced for each pixel, giving an upper bound on render time. This coupled with the fact that the network training and inference take constant time (since $N = 16,384$ for training, and $N = 1024 \times 1024$ for inference, Sect. 5, Alg. 1), we get an accurate estimate of the total time required to render a 1spp frame. Table 2 shows timings to render one spp for the three hair styles, averaged over ten runs. Note that the training & inference timings of our method are the same irrespective of β, as mentioned above.

Figure 11: *We compare 5k spp renderings of our method for* $\beta = 1, 5$ *to 5k spp renderings of Neural Radiance Caching (NRC) [MRNK21], dual scattering [ZYWK08] & a 10k spp path traced reference. We also compare to a version of NRC where all the training paths are unbiased (NRC++). All these methods fail to reproduce the soft look and saturation in hair. Our method with* $\beta = 1$*, which is the most efficient but also the most biased, not only achieves a lower MAP error in most cases, but also reproduces the saturation and soft look.*

6.2. Comparison with previous work

Fig. 11 shows 5k spp renderings of our method for two values of $\beta = 1, 5$ in comparison to 5k spp renders of Neural Radiance Caching (NRC) [MRNK21], dual scattering [ZYWK08] & a 10k spp path traced reference. We also compare to a modified version of NRC, referred to as *NRC++*.

Dual Scattering. Dual scattering is a real-time hair rendering method that efficiently approximates multiple scattering. We use it in the offline context with ray-shooting, as described in their paper. In essence, at the primary path vertex, we shoot multiple rays towards the light source and apply dual scattering for each ray and average their radiance to obtain the final color. Renders using dual scattering are unable to reproduce the soft look and miss a significant component of multiple scattering, especially for light hair (Fig. 11, top row). For darker hair, the renders are slightly better, but still have considerable bias, as depicted by the MAPE values.

NRC. NRC shares a similar approach to ours: They also use a small MLP [Mül21] to efficiently compute global illumination. However, our approach differs in the following:

- We learn on the *final* accumulated radiance, instead of radiance at each path vertex. The latter is useful in surfaces (NRC's target application) with energy quickly degrading deeper in the path, and results in more training data for the same number of paths. However, in hair, this leads to averaging in the network due to excessively long path lengths.
- We train on higher order radiance (E in Eq. (6)), which has much less frequency than the full radiance field. The small MLP is thus able to represent the target signal better (Fig. 3, Sect. 4).

- Our method & formulation provide explicit control over the render's bias & speedup. The bias in NRC however cannot be directly controlled.

As shown in the figure, NRC renders do not faithfully reproduce the color saturation from multiple scattering.

NRC++. NRC takes inspiration from Q-learning and trains the neural network on it's own output. Only a small percentage of their training paths are truly unbiased. We found that using this approach leads to extremely short paths altogether, which are unable to capture higher-order scattering in hair. We thus also compare to a version of NRC where *all* the training paths are traced in an unbiased fashion. We refer to this version as *NRC++*. As shown in Fig. 11, NRC++ is better able to capture the higher-order energy, as compared to NRC which is darker, thanks to better training data. On the other hand, renders of dual scattering result in similar MAPE values as NRC++.

All of the above methods (NRC, NRC++, dual scattering) fail to reproduce the soft look, saturation & multiple scattered component in hair. Our method with $\beta = 1$, which is the most efficient but also the most biased, not only achieves a lower MAP error in all cases, but also reproduces the saturation and soft look.

6.3. Summary

Our method is most beneficial in light hair where it achieves a significant speed & error reduction for small spp. Furthermore, the amount of bias induced by our method can be controlled with β, where larger values result in similar performance & quality as unbiased path tracing. We show that there exists a value of β that achieves the maximum efficiency for a given hair style. The parameter β also gives an upper bound on our method's run-time. This achieves both goals stated in Sect. 1. For darker hair, our method behaves very similarly to unbiased path tracing and has little benefit, suggesting that one could automatically adjust β based on the luminance of the albedo or even turn our method off with dark hairs and $\beta > 9$. Finally, the most biased variant of our method with $\beta = 1$ has a consistently lower error compared to NRC, NRC++ and dual scattering. Visually, our method reproduces the saturation & soft look better than previous approaches (Fig. 11).

7. Conclusions, Limitation & Future Work

We presented an approach to efficiently compute the multi-scattered radiance in hair by learning error between biased short paths and unbiased long paths using an MLP. We described an implementation that efficiently and robustly trains this MLP on the fly, while rendering. We demonstrate the ability of our approach to provide a control over the bias & speedup, specifically allowing to trade unbiasedness for gain run-time and vice versa. We thoroughly analyzed our method and demonstrated that it achieves a speedup of $40\% - 70\%$ with respect to path tracing at the cost of a little bias. We also demonstrated that our method achieves the true color of hair from the get-go, which is useful in look-dev situations. Finally, we compared against an existing general technique that incorporated small MLPs to estimate global illumination (NRC) and with an approximate hair rendering technique for recovering multiple scattering (dual scattering), and showed that our method qualitatively & quantitatively outperforms both.

A limitation of our approach stems from the initial training: training the network from scratch for each frame in an animation setting may lead to temporal artifacts. Although this could be alleviated by choosing a proper β, it needs further investigation. Another limitation is that since the network output is used directly, it may exhibit patterns or colour inaccuracies (Fig. 6 first & last row, Fig. 5 first row). This is less visible when the render converges, but nonetheless needs to be carefully handled. Typical production settings have multiple characters and thus multiple hair grooms in a single frame. For such cases, using a single network with our approach may not be feasible due to network size & learning capacity limitations. One can imagine the use multiple networks for each hair groom and index the corresponding network for different pixels, depending on which groom is visible. However, this remains to be investigated & efficiently engineered.

For future work, we would like to gracefully handle temporal inconsistencies and efficiently render multiple hair-grooms to make our approach truly suitable for production. We would also like to push this technique towards the real-time realm, while maintaining the visual fidelity. Finally, as mentioned in Sect. 3.2, the integral at the last vertex of a path terminated by Russian Roulette can be better estimated, potentially using our network's output, to further reduce variance. This is an interesting direction that can be explored for path tracing of general scenes.

References

[AK90] ARVO, JAMES and KIRK, DAVID. "Particle transport and image synthesis". *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. 1990, 63–66 3.

[BP21] BENAMIRA, ALEXIS and PATTANAIK, SUMANTA. "A combined scattering and diffraction model for elliptical hair rendering". *Computer Graphics Forum*. Vol. 40. 4. Wiley Online Library. 2021, 163–175 2.

[CBTB16] CHIANG, MATT JEN-YUAN, BITTERLI, BENEDIKT, TAPPAN, CHUCK, and BURLEY, BRENT. "A Practical and Controllable Hair and Fur Model for Production Path Tracing". *Computer Graphics Forum*. Vol. 2. 35. 2016, 275–283 2, 6.

[CLZ*20] CHE, CHENGQIAN, LUAN, FUJUN, ZHAO, SHUANG, et al. "Towards learning-based inverse subsurface scattering". *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE. 2020, 1–12 3.

[dFH*11] D'EON, EUGENE, FRANCOIS, GUILLAUME, HILL, MARTIN, et al. "An energy-conserving hair reflectance model". *Computer Graphics Forum*. Vol. 30. 4. Wiley Online Library. 2011, 1181–1187 2, 4.

[FHP*18] FASCIONE, LUCA, HANIKA, JOHANNES, PIEKÉ, ROB, et al. "Path tracing in production". *ACM SIGGRAPH 2018 Courses*. 2018, 1–79 2.

[HHH22] HUANG, WEIZHEN, HULLIN, MATTHIAS B, and HANIKA, JOHANNES. "A Microfacet-based Hair Scattering Model". *Computer Graphics Forum*. Vol. 41. 4. Wiley Online Library. 2022, 79–91 2.

[KM17] KHUNGURN, PRAMOOK and MARSCHNER, STEVE. "Azimuthal scattering from elliptical hair fibers". *ACM Trans. Graph.* 36.2 (2017), 1–23 2.

[KMM*17] KALLWEIT, SIMON, MÜLLER, THOMAS, MCWILLIAMS, BRIAN, et al. "Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks". *ACM Trans. Graph.* 36.6 (2017), 1–11 3.

[LKB*22] LIN, DAQI, KETTUNEN, MARKUS, BITTERLI, BENEDIKT, et al. "Generalized resampled importance sampling: foundations of ReSTIR". *ACM Trans. Graph.* 41.4 (2022), 1–23 4.

[LMH*18] LEHTINEN, JAAKKO, MUNKBERG, JACOB, HASSELGREN, JON, et al. "Noise2Noise: Learning Image Restoration without Clean Data". *Proceedings of the 35th International Conference on Machine Learning, PMLR*. Vol. 80. 2018 5.

[MESK22] MÜLLER, THOMAS, EVANS, ALEX, SCHIED, CHRISTOPH, and KELLER, ALEXANDER. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15 5.

[MJC*03] MARSCHNER, STEPHEN R, JENSEN, HENRIK WANN, CAMMARANO, MIKE, et al. "Light scattering from human hair fibers". *ACM Trans. Graph.* 22.3 (2003), 780–791 2, 4.

[MM06] MOON, JONATHAN T and MARSCHNER, STEPHEN R. "Simulating multiple scattering in hair using a photon mapping approach". *ACM Trans. Graph.* 25.3 (2006), 1067–1074 2, 4.

[MMR*19] MÜLLER, THOMAS, MCWILLIAMS, BRIAN, ROUSSELLE, FABRICE, et al. "Neural importance sampling". *ACM Trans. Graph.* 38.5 (2019), 1–19 3, 5.

[MPG*16] MÜLLER, THOMAS, PAPAS, MARIOS, GROSS, MARKUS, et al. "Efficient Rendering of Heterogeneous Polydisperse Granular Media". *ACM Trans. Graph.* 35.6 (2016) 2.

[MPH*15] MENG, JOHANNES, PAPAS, MARIOS, HABEL, RALF, et al. "Multi-scale modeling and rendering of granular materials". *ACM Trans. Graph.* 34.4 (2015), 1–13 2.

[MRKN20] MÜLLER, THOMAS, ROUSSELLE, FABRICE, KELLER, ALEXANDER, and NOVÁK, JAN. "Neural control variates". *ACM Trans. Graph.* 39.6 (2020), 1–19 3.

[MRNK21] MÜLLER, THOMAS, ROUSSELLE, FABRICE, NOVÁK, JAN, and KELLER, ALEXANDER. "Real-time neural radiance caching for path tracing". *ACM Trans. Graph.* 40.4 (2021), 1–16 3, 5, 11.

[Mül21] MÜLLER, THOMAS. *tiny-cuda-nn*. Version 1.7. Apr. 2021. URL: https://github.com/NVlabs/tiny-cuda-nn 5, 11.

[MWM08] MOON, JONATHAN T, WALTER, BRUCE, and MARSCHNER, STEVE. "Efficient multiple scattering in hair using spherical harmonics". *ACM Trans. Graph.* 27.3 (2008), 1–7 2.

[PBD*10] PARKER, STEVEN G, BIGLER, JAMES, DIETRICH, ANDREAS, et al. "Optix: a general purpose ray tracing engine". *ACM Trans. Graph.* 29.4 (2010), 1–13 5.

[PHVL15] PEKELIS, LEONID, HERY, CHRISTOPHE, VILLEMIN, RYUSUKE, and LING, JUNYI. "A data-driven light scattering model for hair". *Pixar Technical Memo* 2 (2015) 2.

[PJH16] PHARR, MATT, JAKOB, WENZEL, and HUMPHREYS, GREG. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016 3.

[RGH*22] RATH, ALEXANDER, GRITTMANN, PASCAL, HERHOLZ, SEBASTIAN, et al. "EARS: efficiency-aware russian roulette and splitting". *ACM Trans. Graph.* 41.4 (2022), 1–14 3.

[SPJT10] SADEGHI, IMAN, PRITCHETT, HEATHER, JENSEN, HENRIK WANN, and TAMSTORF, RASMUS. "An artist friendly hair shading system". *ACM Trans. Graph.* 29.4 (2010), 1–10 2.

[SSK03] SZÉSCI, LÁSZLÓ, SZIRMAY-KALOS, LÁSZLÓ, and KELEMEN, CSABA. "Variance reduction for Russian-roulette". (2003) 3.

[Vea98] VEACH, ERIC. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998 3, 5.

[VK16] VORBA, JIŘÍ and KŘIVÁNEK, JAROSLAV. "Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925912. URL: https://doi.org/10.1145/2897824.2925912 3.

[VKJ19] VICINI, DELIO, KOLTUN, VLADLEN, and JAKOB, WENZEL. "A learned shape-adaptive subsurface scattering model". *ACM Trans. Graph.* 38.4 (2019), 1–15 3.

[WPW89] WYMAN, DOUGLAS R, PATTERSON, MICHAEL S, and WILSON, BRIAN C. "Similarity relations for the interaction parameters in radiation transport". *Applied optics* 28.24 (1989), 5243–5249 3.

[XWM*20] XIA, MENGQI, WALTER, BRUCE, MICHIELSSEN, ERIC, et al. "A wave optics based fiber scattering model". *ACM Trans. Graph.* 39.6 (2020), 1–16 2.

[YSJR17] YAN, LING-QI, SUN, WEILUN, JENSEN, HENRIK WANN, and RAMAMOORTHI, RAVI. "A BSSRDF Model for Efficient Rendering of Fur with Global Illumination". *ACM Trans. Graph.* 36.6 (2017) 2.

[ZRB14] ZHAO, SHUANG, RAMAMOORTHI, RAVI, and BALA, KAVITA. "High-order similarity relations in radiative transfer". *ACM Trans. Graph.* 33.4 (2014), 1–12 3.

[ZW07] ZINKE, ARNO and WEBER, ANDREAS. "Light scattering from filaments". *IEEE Transactions on Visualization and Computer Graphics* 13.2 (2007), 342–356 2.

[ZYWK08] ZINKE, ARNO, YUKSEL, CEM, WEBER, ANDREAS, and KEYSER, JOHN. "Dual scattering approximation for fast multiple scattering in hair". *ACM Trans. Graph.* 27.3 (2008), 1–10 2, 5, 11.

[ZZW*22] ZHU, JUNQIU, ZHAO, SIZHE, WANG, LU, et al. "Practical level-of-detail aggregation of fur appearance". *ACM Trans. Graph.* 41.4 (2022), 1–17 3.