

CGHair: Compact Gaussian Hair Reconstruction with Card Clustering

Supplementary Material

6. Additional Results

6.1. Computational time analysis

We conduct an ablation study comparing the computational time required by our method with the current state-of-the-art techniques. All experiments are trained on the same setup using a single NVIDIA RTX A4500 GPU.

Strands Reconstruction We compare the strand reconstruction time of our approach with the current state-of-the-art methods in Tab. 6. As shown, our proposed method achieves approximately a fourfold speedup over existing techniques.

Table 6. Comparison of the strand reconstruction time.

Model	Recon. Time	Ratio
GaussianHairCut	11h	4.14
GaussianHair	8h	3.01
Ours	2h40m	1.00

Total end-to-end Reconstruction Time Analysis In Tab. 7, we compare the end-to-end reconstruction time of our method to current state-of-the-art approaches. Our method demonstrates approximately a threefold speedup over previous techniques.

Table 7. Comparison on Time for End-to-End Reconstruction.

Model	Recon. Time	Ratio
GaussianHairCut	15h	3.00
GaussianHair	10.5h	2.10
Ours	5h	1.00

6.2. Strands reconstruction ablation

To further evaluate our strand generator, we also conduct the same ablations done in Sec. 4.3 on real-captured hairstyles. These hairstyles provide a more realistic and challenging test set, enabling us to assess the generalization and effectiveness of our approach more accurately in real-world scenarios as it can be seen in Fig. 10.

6.3. CGHair

Memory Footprint Analysis. We present a quantitative analysis of the memory cost of the file size and appearance features of our method in Tab. 8.

Table 8. Comparison of storage cost.

Model	Appearance Features		File size	
	Size (Mb)	Ratio	Size (Mb)	Ratio
GaussianHairCut	544	464	93	77.51
GaussianHair	505	431	90	75
Ours	1.17	1.00	1.20	1.00

Quantitative comparisons. In Tab. 9, we present a quantitative comparison of the appearance quality of different methods using three common metrics: PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), and LPIPS (Learned Perceptual Image Patch Similarity). When comparing the performance of the three methods—GaussianHairCut, GaussianHair, and Ours—across three hair types (Curly, Short, and Long), we observe that GaussianHair performs the best, while our method (Ours) exhibits competitive performance, with slightly lower PSNR and SSIM values than GaussianHair, yet still outperforms GaussianHairCut in most cases. Notably, our method achieves this superior rendering performance with only a fraction of the parameters used by existing state-of-the-art methods. Specifically, we have reduced the parameter count by over 200 times. This demonstrates the efficiency of our approach, achieving high-quality hair strand reconstruction with a drastically smaller model.

Qualitative ablations. We show the rendered images using our ablated models in Fig. 11. Here, we have carefully designed the rendering scheme for a synthetic hairstyle, where each strand of hair is rendered in multiple segments with different colors, providing an extremely challenging test. This design allows us to sensitively evaluate the model’s ability to preserve fine-grained strand continuity and appearance consistency under complex spatial and color variations. As shown in Fig. 11, we present four unseen views spaced relatively far apart.

Consistent with the metrics in Tab. 2, *W/o card clustering* exhibits the worst rendering performance. The *Single-SH* model works well on most common monochromatic hair datasets, but it generates many artifacts in this challenging case, for both geometry and appearance. Although *Multi-SH* achieves high rendering metrics, some perspectives exhibit noise, as seen in the last row. As *Unique* is designed analogously to GaussianHair, it demonstrates good rendering performance too; however, the model has a significantly larger number of parameters. In contrast, our method has only 1/200th of the parameters, offering an excellent performance-to-parameter trade-off.



Figure 10. We show the ablations of our strand generator on several real captured hair styles.

Table 9. **Quantitative comparison on appearance.** Across three hair types (Curly, Short, and Long) from GaussianHair dataset.

Method	Curly			Short			Long		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
GaussianHairCut	30.07	0.9225	0.0565	31.03	0.9041	0.0774	27.78	0.8719	0.1010
GaussianHair	31.56	0.9290	0.0466	32.07	0.9020	0.0597	29.12	0.8791	0.0749
Ours	30.34	0.9153	0.0528	31.15	0.8928	0.0696	28.60	0.8696	0.0815

6.4. Additional Results of Various Hairstyles and free view visualizations

We provide additional qualitative results for various hairstyles on both the MonoHair [71] and the H3DS [53] datasets. As shown in Fig. 12, we show our automatically generated results on 3 MonoHair hairstyles. Our results are better than GaussianHaircut, while MonoHair can preserve more details, which is benefited from a manually tuned hair growing algorithm. As shown in Fig. 13, we also provide more appearance rendering results of both our method and GaussianHaircut. Our method achieves comparable rendering ability with a much lower memory footprint. In Fig. 14, we showcase our generated strands of H3DS data, where GaussianHaircut failed to reconstruct such a hairstyle.

Visualizations of Hair Card Textures. As stated in Sec. 3.2, our method creates hair cards from the grouped strand clusters, and then the hair card textures are generated by ‘projecting’ the strands onto the hair card mesh faces. We visualize several resulting textures in Fig. 17 of a real-captured hairstyle *big wavy*, from the MonoHair [71] dataset. These textures are then used to cluster the hair cards into similar groups further, to finally generate our shared Gaussian texture codebook.

6.5. Dynamic Simulation.

As shown in Fig. 16, we show the renderings of the simulated hair strands, for a synthetic hairstyle, the real captured hairstyle *long* from GaussianHair dataset and the hairstyle *big wavy* from MonoHair dataset. Note that we adopt the simulation results from the CG engine and apply them to drive our Gaussian hair strands; in this way, we render the strands in the standard Gaussian rendering framework.

7. Limitation and discussion.

As a compact Gaussian-based hair representation that can be reconstructed from multi-view static hair images at a superior memory compression ratio, our work still has several limitations:

Failure Cases. As an image-based hair reconstruction method, our fully automated, efficient hair strands reconstruction method cannot handle extremely challenging hairstyles, like afro-textured or tightly braided hairstyles. As a general limitation, the previous automated 3D Gaussian-based method, e.g., GaussianHaircut [78], failed on such challenging cases. While other works, such as GaussianHair [36] or the NeRF-based method Mono-

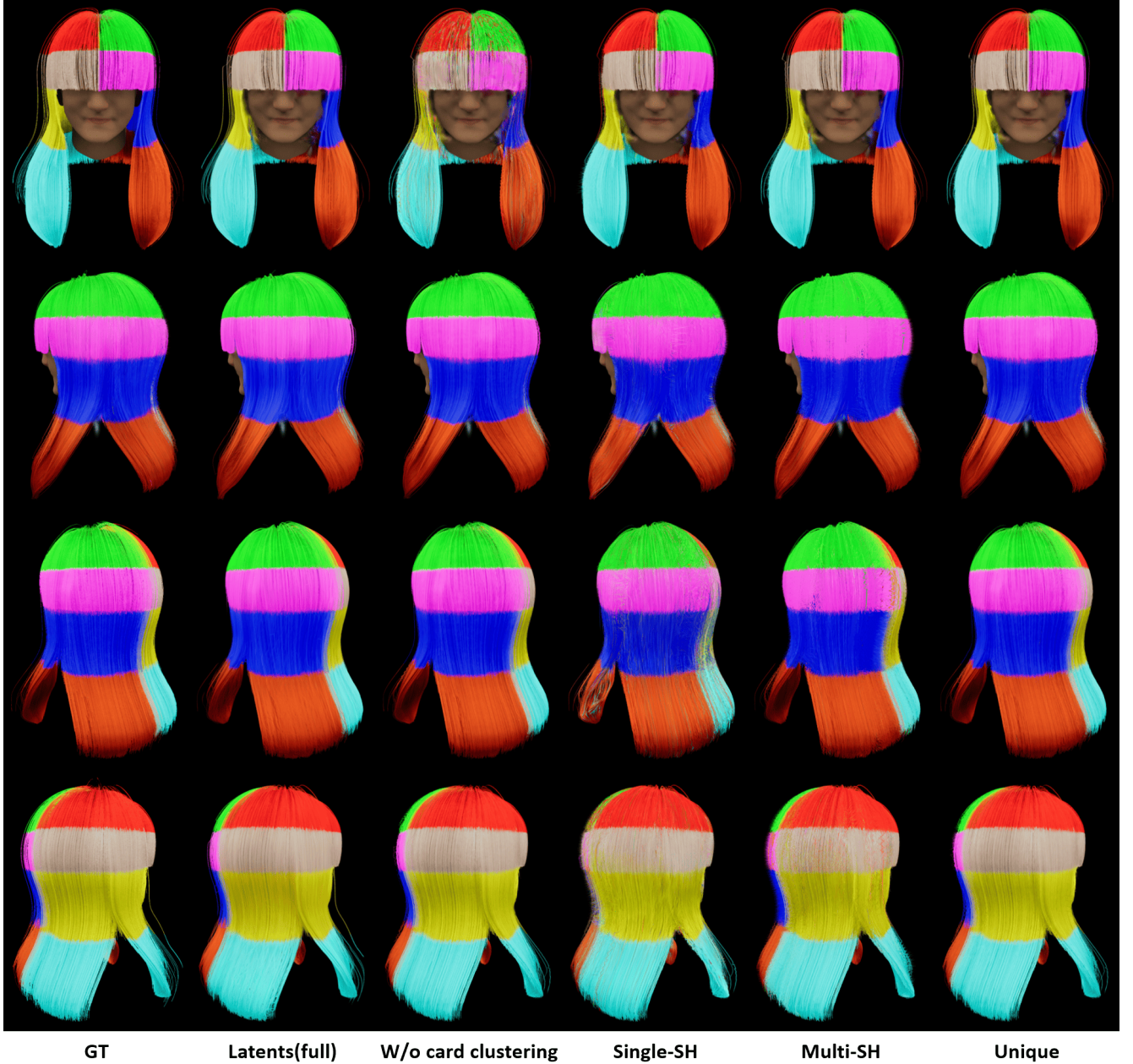


Figure 11. Qualitative ablation results using a carefully designed synthetic hairstyle, illustrating the algorithm’s robustness in complex scenarios where hair exhibits multiple color tones.

Hair [71], also failed, even though they require more manual tuning for better results. The main reason is that the complex, intricate hair structure cannot be observed from external views. A possible direction is to collect more complex synthetic hairstyles or real CT hairstyles [59] to provide better priors for inner hair structure.

Dynamic Scenario. As our method maintains the explicit hair strands, using hair cards only for clustering and reduc-

ing the storage and redundancy of Gaussian-based hair representations, it naturally enables strand-based simulations in CG engines. As shown in Fig. 15 and Fig. 16, the resulting strand animation from the CG engine can further drive our CGHair. However, the current model is trained only on static monocular captures; the animated results, while keeping visually reasonable hair geometry, may inevitably exhibit artifacts around occluded regions (e.g., occluded body part in Fig. 15) that were never observed during training.

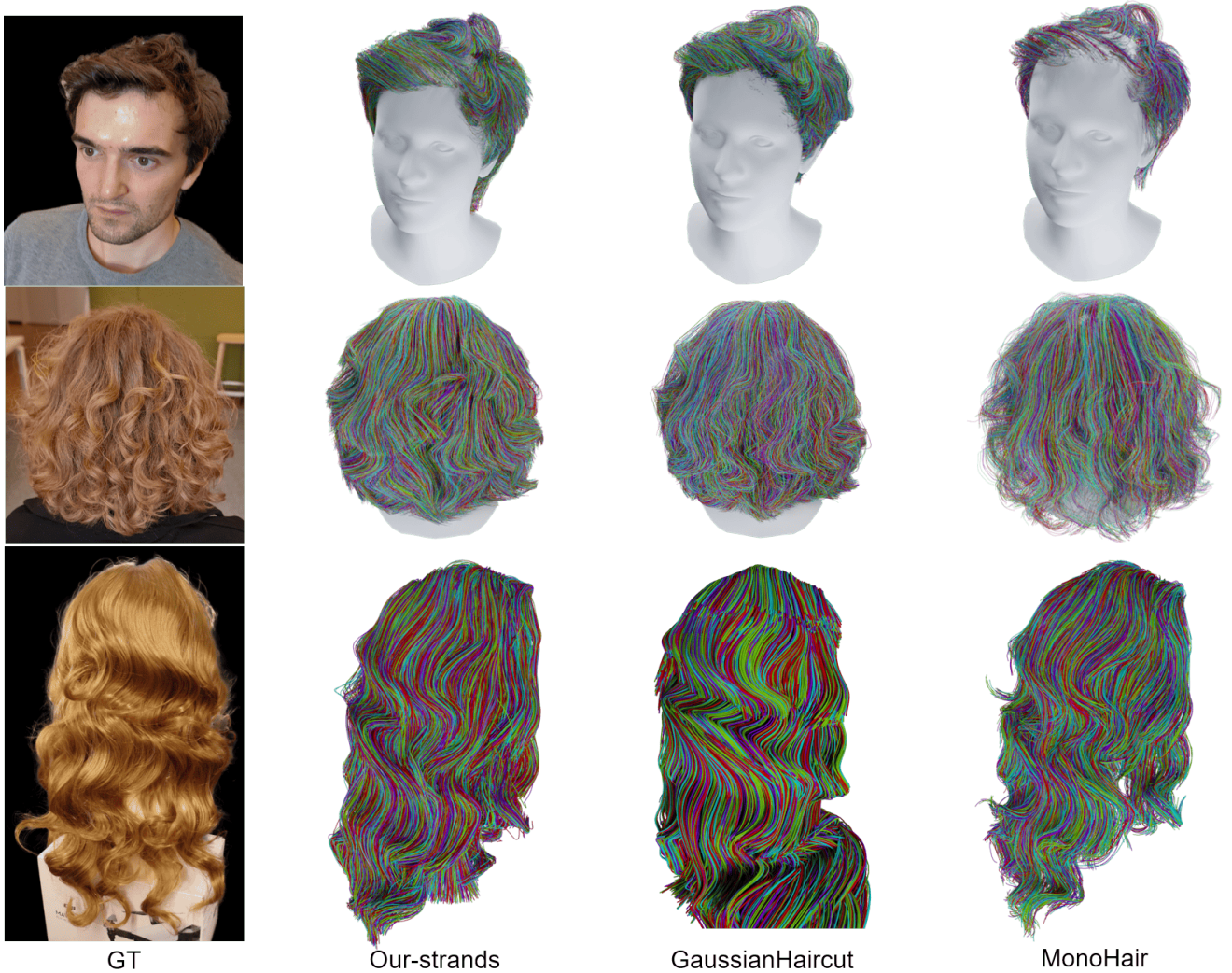


Figure 12. More hairstyles from the MonoHair dataset. We compare our reconstructed strands against MonoHair and GaussianHaircut.

Moreover, since Gaussian representations can easily overfit the visible input views, the hair appearance from unseen viewpoints may become less natural. Equipping the current framework with current dynamic techniques and training with dynamically captured multi-view hair video data is a promising direction to mitigate these limitations.

Visual Details Loss. While our shared Gaussian appearance codebook, driven by card-based hierarchical clustering, achieves over a 200x reduction in the memory footprint of the Gaussian representation, it comes with a trade-off: a slight loss of high-frequency hair details in the appearance rendering. This degradation primarily stems from our current hard-clustering strategy, which remains static once initialized. Therefore, exploring a dynamic or differentiable clustering mechanism that can be jointly optimized during

the training process represents a highly desirable direction for future work.



Figure 13. More Hairstyles appearance rendering comparisons. We compare our appearance rendering results against GaussianHaircut on 3 hairstyles from the MonoHair dataset and one from the H3DS dataset.

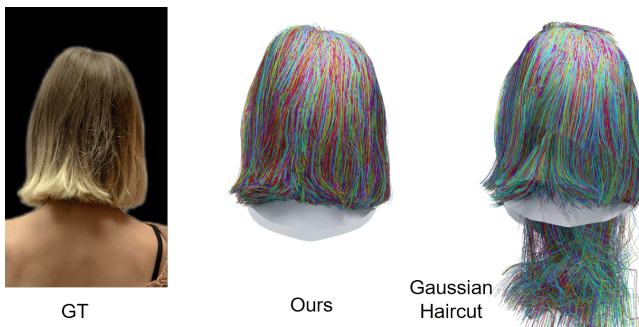


Figure 14. We also compare our reconstructed strands with GaussianHaircut on the H3DS hairstyle; note that GaussianHaircut failed.



Figure 15. Animated hair: On the left, original reconstruction, on the right, random timestamp frame from a strand-based animation.



Figure 16. Our explicit strand representation naturally enables strand-based dynamic simulation for a synthetic hairstyle (top), a real captured hairstyle from GaussianHair dataset (middle), and the other one from MonoHair Dataset (bottom). The simulated Gaussian strands are rendered using the standard Gaussian rendering pipeline. Please refer to the project webpage for a better visualization of the dynamic motions.

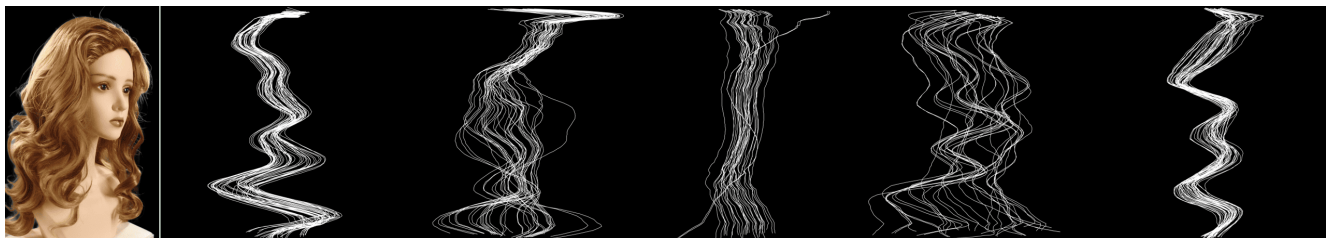


Figure 17. We show several example representative strand textures for grouped strands, which are represented by a hair card. The results are reconstructed from the hairstyle *big wavy* from the Monohair dataset.



Figure 18. Visualization of our additional free-view renderings of real-captured hairstyles.