EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Dynamic Gaussian Visualizer

Daniel Eskandar
César Díaz Blanco

May 3, 2026

GitHub Repository

Practical Machine Learning Summer 2024
Graphics, Computer Vision and Machine Learning

Dynamic Gaussian Visualizer

Daniel Eskandar
César Díaz Blanco

GitHub Repository

Figure 1: Our visualizer allows for multiple head avatars can be displayed and edited.

Motivation

We present a head-avatar-focused interactive Gaussian splatting visualizer with support for dynamic scenes, FLAME models, hair editing, among other features. We take advantage of the visual quality, controllability, and efficient rendering of Gaussian splatting [KKLD23] adapted to the field of virtual humans. In this project, we implemented features which enable the user to: visualize dynamic scenes, do editing of head-avatars, and compare head-avatars.

The dynamic features include dynamic gaussians by loading new position, scale and rotation attributes per frame; controllability of head-avatars through a FLAME [LBB+17] mesh model as shown in GaussianAvatars [QKS+23]; and a view which shows the gaussians' axes. The editing features include toggling on and off the gaussians corresponding to the head or hair, coloring any gaussian, removing hair gaussians as to change the strand length, curling the hair with controllable amplitude and frequency, and saving the new gaussian splatting model file with edits. Lastly, the comparison features lets the user visualize head-avatars side by side and swap the hair between avatars.

Base visualizer

Our visualizer was built from the publicly available Tiny Gaussian Splatting Viewer under an MIT license. This visualizer and our features are built

on OpenGL and thus it is possible to use without the need of an NVIDIA GPU. The reasoning behind using this visualizer is that it's the leanest and most compatible implementation we could find. This is shown by the myriad number of features we implemented without major drawbacks. On the other hand, we were not able to implement GPU routines to speed up aspects of our code like sorting the gaussians, displaying the FLAME gaussian model and changing its parameters, or computing the curls' displacement on the fly. Nonetheless, even with our biggest model with 620k hair gaussians and 12207 head gaussians, the dynamic updates are smooth enough.

Dynamic features

1 Dynamic Gaussians with frame controller

We enable dyanamic hair through a `.npy` file to load a three-dimensional array with the mean, rotation, and scale values of the gaussians along the sequence. The ordering is such that the first dimension has as many entries as frames, the second dimension has as many entries as hair gaussians, and the third dimension is the concatenation of the mean, rotation, and scale in that order. A script to pack these frames and more details on its usage are described in the project's repository. To reduce the number of read operations we load all the parameters into a single array instead of in several files.

2 Interactive FLAME model

We support the use of head avatars with FLAME [LBB+17] properties as seen in GaussianAvatars [QKS+23] by loading the respective folder with the hair non-FLAME ply file, face FLAME ply file, and FLAME mesh ply file. A FLAME control window lets the user change the parameters of the neck, jaw, and eyes as well as the first five principal components of the face expressions. Each hair strand is attached to the scalp by finding the nearest FLAME vertex. When the FLAME parameters are adjusted, the hair moves along with the head by applying the Linear Blend Skinning (LBS) transformation of the corresponding closest vertex to each strand.

3 Axes view

To ease debugging of the hair gaussians, we implemented an axes view which lets the user determine if the hair gaussians of a strand are well oriented and connected. This was done through the addition of a fragment and vertex shader, as the original renderer uses triangle primitives while this view requires rendering with line primitives.

Editing features

4 Show or hide hair and head

To let the user identify whether the segmentation and colors are appropriate, we implemented the toggling of either the head or hair sections.

5 Coloring

In the same spirit as the previous feature, the coloring feature is meant to correct the colors of the hair which may merge colors from the hair and the neighboring region (shoulders, back, scalp). There are two modes to this feature:

- Quick Segmentation: colors all the gaussians from the head or hair in a single color.
- Coloring by clicking: enabled for both hair and head segmentation by right-clicking on the region to be painted over. The color can be either selected from an RGB menu or by left-clicking over a gaussian whose color is to be used.

6 Hair cutting

This feature is only enabled for the hair gaussians by right-clicking on the region to be cut. When it comes to each strand, if an intermediate gaussian is selected, then the subsequent gaussians are deleted as well as to not have flying strands.

7 Curly hair control

For this feature, we take each of the hair strands and apply a curly hair effect by displacing the gaussians along the axes perpendicular to the strand by a sine and cosine distance from its original mean. The frequency and amplitude of these sine and cosine functions can be changed by the user in the visualizer to achieve the desired curliness.

To ensure the gaussians remain connected we compute their orientation by their rotation matrices' columns which represent the gaussians' tangent, normal and binormal vectors. We optimized this as much as possible by exploiting the embarrassingly parallel quality of this problem as each of the strands is independent of each other. However, since we only focused on a CPU implementation this means we could only leverage the linear algebra optimizations of the `numpy` library.

To further speed up this process, we modified the renderer code to take rotation as matrices instead of quaternions. The renderer ultimately needs the rotations as matrices to get the gaussians' covariance. We keep the implementation separate from the `main` branch since other features rely on the rotation being kept as a quaternion.

We also let the user pre-compute the mean, rotations, and scale of each of the hair gaussians and then load it in the visualizer. For more information on how to use the script to do so please refer to the repository.

8 Export ply file with edited color, cut, and curls

In this feature, we take the array of the gaussians' mean, scale, and rotation that is sent to the renderer and pack it as the standard `.ply` file for gaussian splatting scenes. By doing so we make sure that the user's edits in color, cutting, and curls are exported.

Comparison of avatars features

9 Display multiple heads

Our visualizer supports the loading of multiple gaussian splatting scenes including head avatars. The avatars are displayed horizontally from left to right as shown in the banner image. This feature assumes the scenes share the same scale and allows the user to compare the avatars as well as modify each of them individually with the editing features by left-clicking on the head avatar which is to be edited.

10 Hair swapping

This feature assumes that the scenes share the same scale and enable the user to swap the hair of an avatar with that of any other loaded avatar. Furthermore, the hair swapping preserved the edits in color, cutting, and curls.

References

[KKLD23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[LBB+17] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017.

[QKS+23] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. *arXiv preprint arXiv:2312.02069*, 2023.

Conclusion

We have introduced a Dynamic Gaussian Splatting Visualizer tailored for head avatars, enabling interactive editing, dynamic scene rendering, and avatar comparison. By leveraging the efficiency of Gaussian splatting and the flexibility of FLAME models, our tool provides users with fine-grained control over head and hair modifications. Despite being CPU-based, our implementation ensures smooth performance even with large-scale models, while offering a range of features such as hair styling, color correction, and real-time adjustments.

Future work could focus on GPU acceleration for computationally intensive tasks such as Gaussian sorting and deformation, as well as expanding compatibility with additional avatar representations. We believe our visualizer provides a valuable foundation for researchers and developers working on virtual human modeling, animation, and real-time rendering.