Photo-Realistic Composite Faces

An Editor of Photo-Realistic Faces Using 3D Gaussian Splatting and Mesh Blending

MSc Thesis, Computer Graphics & Visualisation

Renzo Russel

Delft University of Technology

TU Delft

Photo-Realistic Composite Faces

An Editor of Photo-Realistic Faces Using 3D Gaussian Splatting and Mesh Blending

by

Renzo Russel

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 28, 2025 at 09:00 AM.

Student number:	4557719	
Project duration:	October 28, 2024 – August 28, 2025	
Thesis committee:	Dr. M. Weinmann,	TU Delft, supervisor
	Dr. X. Zhang,	TU Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

I would like to begin by thanking my parents, Clark and Jessica Russel, who, despite my many side quests and delays, have always encouraged me. Without their support, I would never have made it through my studies. Their belief in me has carried me through many challenges.

A very special thank you goes to my girlfriend, Maria Alexe, who has spent the past year patiently listening to my endless ranting and raving about Gaussians. She stood by me even in the most stressful times, making sure I slept and ate. Your patience and understanding mean more to me than I can put into words.

I am also deeply grateful to my supervisor, Michael Weinmann, for his guidance, support, and constructive feedback throughout this work. His encouragement has been invaluable. I'm also thankful to Amir Zaidi for (unofficially) supervising me. His insightful ideas and suggestions greatly helped shape and improve this thesis. Without his insistence to "just try it out and see," half of this thesis might never have existed.

Finally, I would like to thank all of the BESTies who made every step of my journey as a student more exciting and fun. Although my time as a student would almost definitely have been half as long without you, it would certainly not have been half as meaningful. Work hard, play harder.

Renzo Russel
Delft, August 2025

P.S. Dani you're next...

Contents

1 Introduction

In today's society, digital technologies play a central role in how people communicate, work, and interact with the world. Visual media, in particular, have become indispensable across domains such as entertainment, commerce, education, healthcare, and social interaction. As the demand for realistic digital representations continues to grow, applications ranging from autonomous navigation and virtual design to the use of lifelike avatars in rehabilitation and online environments increasingly rely on high-quality digital content. Among these, the realistic depiction of human faces holds special significance, since faces are central to human interaction and recognition.

As 3D graphics become more realistic, it becomes harder for 3D face models to keep up. Human faces must not only render convincingly, but also move naturally in order to avoid breaking the illusion of realism. This challenge is amplified by the human brain's extraordinary sensitivity to facial details, where even subtle inconsistencies can lead to discomfort or distrust, a phenomenon described by the uncanny valley effect [7].

Addressing these challenges is a complex task for digital artists, since creating and animating high-fidelity 3D faces manually requires both technical skill and significant time investment. To overcome these limitations, the industry increasingly relies on captured data from real-world actors in the form of 3D facial scans and facial motion capture [23, 31, 9]. These data-driven techniques have proven highly effective at producing realistic results and are widely adopted in film, gaming, and virtual production. However, they come with several drawbacks: they require expensive hardware setups, are time-consuming to process, demand substantial rendering resources to achieve the highest levels of visual quality, and often lack flexibility when adapting captured assets to new contexts.

Recent work has yielded the breakthrough 3D Gaussian Splatting (3DGS) [12], which steps away from conventional mesh-based rendering, instead using a learning-based volumetric representation based on 3D Gaussians typically generated from real-world images. This scene representation is adept at modelling complex geometric and lighting characteristics, while still allowing scene visualisation at high frame rates. Additionally, thanks to the learning-based approach used in the digitisation process, the produced scenes are photo-accurate.

An emerging use case for 3D Gaussian Splatting [12] is in capturing 3D photo-realistic human faces [21, 30, 28, 17]. These methods are very adept at creating photo-realistic 3D heads and, by using an underlying mesh structure, support animating the heads. Such an animatable head is referred to as a Gaussian head avatar.

One crucial limitation of these methods is that they restrict which faces artists can work with by requiring the capture of new footage and going through the expensive and time-consuming process of generating a new Gaussian head avatar from this footage. To address this, we propose a novel method to create new animatable Gaussian head avatars by selectively combining semantic features from an existing database of source Gaussian head avatars. Our method makes it possible to reuse avatars to easily and quickly create new ones. Thereby, this thesis aims to ***examine the feasibility of selectively***

combining 3D Gaussian head avatars to create a novel and convincing photo-realistic head avatar for 3D animation. To demonstrate our approach, we develop a composite editor for 3D Gaussian head avatars and conduct a study with human participants tasked to distinguish randomly edited composite avatars from unedited source avatars. In summary, the key contributions of this thesis are:

- A novel method to create Gaussian head avatars by combining existing Gaussian head avatars. Our method makes it possible to create new avatars without any of the usual costs associated with creating a 3DGS scene.
- A modified version of GaussianAvatars' [21] optimisation method, addressing interference from Gaussians bound to opposing sides of the underlying mesh by temporarily inserting opaque Gaussians inside the mesh during the optimisation process. This makes it possible to use front-view-only footage to generate Gaussian head avatars suitable for our editing method.
- Two Gaussian post-processing methods: one to smooth transitions between Gaussians to mask artefacts between two facial regions, and another to adjust colours for consistent skin tones, enabling composite avatars from sources with different skin tones.
- A performant and interactive editor for photo-realistic head avatars based on mesh-bound Gaussians, capable of selectively combining Gaussians and face shapes from multiple source avatars.
- An evaluation of our method according to a study evaluating the perceived authenticity of our composite avatars with human participants, as well as a real-time performance analysis of our editor to evaluate our method for use in interactive editing.

2
Background

In this chapter, we explain relevant background knowledge. This includes developments in the domain of 3D scene reconstruction, also including the 3D Gaussian Splatting approach [12] that has gained a lot of attention in recent years. Furthermore, we discuss mesh-bound Gaussian approaches that have been successfully used for face representation. In particular, GaussianAvatars [21], which this thesis directly builds on.

2.1. 3D Scene Reconstruction

Traditional 3D scene reconstruction pipelines typically use Structure-from-Motion (SfM) and multi-view stereo approaches to explicitly model scene geometry in terms of a surface mesh from image data [25, 26, 33]. These methods integrate well with standard computer graphics rendering engines and can achieve accurate geometric reconstructions. However, due to their mesh-based representations, they struggle to capture the really fine geometric details and due to typically only storing a single colour value per surface point, they additionally cannot handle complex view-dependent lighting effects crucial for photo-realism.

With the advent of Neural Radiance Field [19] (NeRF), learning-based scene reconstruction has steadily gained popularity. In contrast to the traditional approach, NeRF [19] scenes are represented implicitly by a neural network acting as a function mapping 3D coordinates and viewing directions to colour and density values. The continuous radiance field represented by this function is trained using image-based loss, which allows it to create high-fidelity renders. The biggest downside to NeRF [19] is that rendering is performed by querying the neural network many times per pixel. This is very costly, making it ill-suited for real-time use.

2.2. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [12] is a 3D scene reconstruction technique which also represents scenes as a radiance field. However, instead of a neural network, a 3DGS scene is made up of many ellipsoids called 3D Gaussians, each representing a discrete part of a continuous radiance field. As a distinct set of primitives, these Gaussians can be rendered orders of magnitude faster than NeRF [19], achieving high-fidelity novel-view synthesis at performant frame rates.

Each Gaussian is described by its position μ, opacity α, colour c, and shape defined by a 3D covariance matrix Σ. In practice, the covariance matrix is derived from the rotation and scaling matrices r and s of a 3D ellipsoid according to $\Sigma = rss^Tr^T$. The colour is represented using Spherical Harmonics coefficients [22] to capture view-dependent lighting effects, e.g. specularity.

2.2.1. Scene Generation

Kerbl et al. [12] create a 3DGS scene by taking real-world captured footage as their ground-truth. The Gaussians used to represent the scene are initialised based on points obtained from Structure-from-

Motion [25] or randomly. The parameters of these Gaussians are then modified to better represent the scene in terms of leading to synthesised views that match the ground-truth photographs given for the respective camera configurations. This is done iteratively according to a gradient descent optimisation technique [24]. Additionally, a periodic densification step is involved to better adapt the number of Gaussians in the scene as needed. This is done by cloning or splitting existing Gaussians (Figure 2.1).

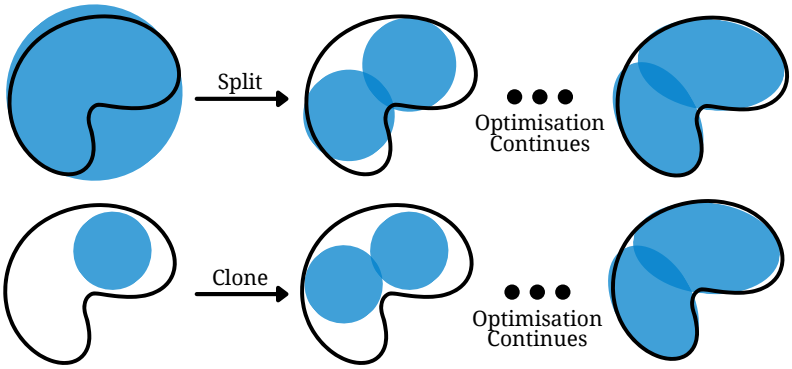


Figure 2.1: Densification of Gaussians based on a figure from [12].

2.2.2. Rendering Gaussians

Gaussian Splatting rendering is equivalent to marching a ray through a scene and adding the colour and opacity of each Gaussian that intersects that ray (see Figure 2.2) until saturation is reached. The colour C of a pixel is computed from the ordered set $\mathcal{N}$ of points along this ray according to:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i T_i$$

with colour c_i and opacity a_i of each point i along the ray and transmittance

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

modelling the exponential loss of light along the ray from all the points preceding i.

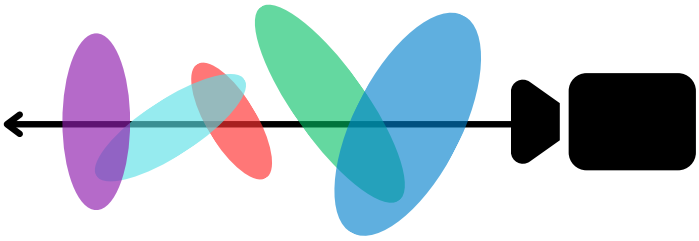


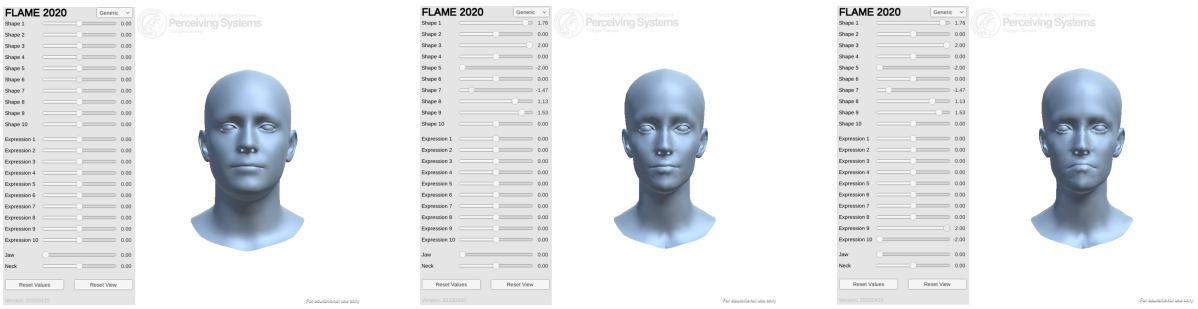
Figure 2.2: Illustration of ray marching used in Gaussian splatting rendering.

2.2.3. Novel-View Synthesis

Because Gaussians are 3D objects in space, it is possible to reposition the camera around the scene at will. This allows for the rendering of novel views that are not contained in the set of photographs used for the inference of the 3DGS scene. For camera configurations similar to certain configurations included during the capture process, the synthesised views are more accurate than for views that more strongly deviate from respective configurations.

2.3. 3D Morphable Face Models

A 3D Morphable Face Model (3DMFM) [1, 4] is a statistical model that represents the shape and appearance of human faces in 3D. Such models enable the generation of 3D face meshes by adjusting a set of linear parameters (e.g. via sliders) to manipulate facial shape. Because these meshes have a standard mesh topology, they can be used to simulate mesh deformation. Conventional 3DMFMs rely on principal component analysis (PCA) [10] to define a latent space for 3D face shapes from a set of facial scans. One prominent example is FLAME [15], which employs a learned linear latent space consisting of 300 shape, 100 expression, and 6 pose parameters. Given an input image of a face, FLAME [15] can estimate shape and expression parameters, enabling 3D face reconstruction. Explicitly separating shape and expression parameters makes independent manipulation of these attributes possible (see Figure 2.3). This separation also makes it possible to animate a face from driving footage by transferring extracted expression parameters to a different set of shape parameters.



(a) Neutral FLAME face

(b) FLAME mesh with modified shape parameters

(c) FLAME mesh with modified shape and expression parameters

Figure 2.3: Online interactive FLAME viewer accessible at https://flame.is.tue.mpg.de/interactivemodelviewer.html.

2.4. Mesh-Bound Gaussians

Gaussians, by their unstructured nature, don't lend themselves well to the kind of surface deformation required for animation. This limitation has been addressed in various works by binding the Gaussians to an underlying mesh [6, 30, 5]. The exact way this is done differs, but the core idea is the same: by binding each Gaussian to a parent mesh primitive (usually a triangle), it is possible to transitively update the attributes of a Gaussian according to its parent's deformation. This is typically done by expressing the scale, rotation, and mean of a Gaussian as a function of its parent's 3D coordinates. The strategy used to bind Gaussians to a mesh differs, but in general, the mesh should match the underlying shape of the object the Gaussians are being bound. As a result, the meshes used are typically extracted from the same images used to optimise the Gaussians. A common approach is constructing a mesh, binding Gaussians to it, and jointly optimising the Gaussians and the mesh itself to more accurately match the ground-truth images [5, 6, 21]. Using this approach with video footage of a moving subject requires the generation of a new mesh whenever the subject moves. This is a problem because mesh extraction methods typically don't produce the same mesh topology, which means the Gaussian bindings cannot be used across meshes. The use of 3DMFMs can solve this problem due to their use of a standard mesh topology.

2.4.1. Gaussian Head Avatars

A popular use case for mesh-bound Gaussians is to create high-fidelity, animatable, human faces [21, 30, 28, 35]. Due to the dynamic nature and tendency for occlusion (e.g. the opening/closing of eyes and mouths) of human faces, it's desirable to optimise the Gaussians on footage of a moving face for a more accurate result. Unfortunately, a new mesh is required whenever the subject moves, otherwise the mesh won't accurately match the subject. However, if the topology of the mesh changes, as is the case in many mesh extraction methods, the Gaussian bindings won't match the mesh either. A solution to this problem is to use a 3D Morphable Face Model for its ability to generate a mesh with standard topology from images. This makes it possible to use footage of a moving subject to generate a mesh and mesh-bound Gaussians.

GaussianAvatars

GaussianAvatars [21] is a method which uses FLAME [15] to create photo-realistic head avatars. Their avatars are animatable through modification of the FLAME expression parameters. The mesh manipulation used by GaussianAvatars only extends to expression transfer. We add to this by introducing methods for shape transfer and texture transfer by facial region. As we build directly on GaussianAvatars, their specific method is covered in more detail below.

Qian et al. [21] define each Gaussian in relation to a single parent triangle on a mesh. The centre point μ, rotation r, and scaling s of a Gaussian are all defined relative to the tangent space defined by its parent triangle. To render the Gaussians, they must first be converted to world-space according to:

$$
\begin{aligned}
r' &= Rr \\
\mu' &= kR\mu + T \\
s' &= ks
\end{aligned}
$$

Where T denotes the triangle centroid, R represents the rotation matrix describing the orientation of the triangle, and the scalar k represents the size of the triangle. During the mesh updates, the parameters T, R, and k for each parent triangle are updated, which then updates their respective Gaussians' parameters in world space.

Gaussians are optimised directly onto a mesh. Each triangle of the mesh is initialised with a single Gaussian bound to it. The tangent-space parameters are then optimised using gradient descent [24] as normal. Gaussians with very low opacity or scale are periodically removed, with the restriction that every triangle must have at least one Gaussian bound to it. This is because Gaussians created by densification inherit their binding from another Gaussian; thus, if a triangle loses all its Gaussians, it will never get them back.

Gaussians that are far from or much larger than their parent triangle react drastically to small changes in the triangle. To prevent this, Qian et al. [21] introduce additional loss terms for the relative scale and distance of a Gaussian to its parent triangle. The result is Gaussians, which approximate a surface by acting as a kind of "volumetric texture" for a mesh.

3

Related Work

Gaussians by themselves, as a disorganised point cloud, lack structure. This makes them ill-suited for conventional mesh-based animation and editing techniques. In this chapter, we provide an overview of some works addressing the lack of underlying structure in Gaussians, as well as look at some approaches to editing Gaussian scenes.

3.1. Adding Structure to Gaussians

SuGaR [6] creates an animatable mesh-bound Gaussian model from ground-truth footage. The approach used by the authors is to add a regularisation term during optimisation, which favours thin flat Gaussians placed close to the surface. These surface-aligned Gaussians are then used to synthesise a mesh using Poisson reconstruction [11]. Lastly, they bind the Gaussians to the mesh, then jointly optimise both the mesh and the Gaussians. However, SuGaR does not produce a standard mesh topology, which is needed to maintain binding across different meshes both during optimisation and during editing. In contrast, the approach presented in this thesis will overcome this problem.

GaussianFrosting [5] approaches the problem of surface reconstruction by treating Gaussians not as a thin flat "2D-like" surface, but instead as a "frosting" layer of variable thickness. This layer is thicker for volumetrically complex features (e.g. fur) and thinner for well-defined flat surfaces (e.g. a table). The frosting layer is defined for each triangle by taking its vertex normals and applying a positive and negative shift to get 6 points defining a triangular prism. A Gaussian is then defined relative to its parent prism, with its centre expressed in barycentric coordinates to ensure that it stays inside the prism. This results in mesh-bound Gaussians that perform well for both flat surfaces and volumetric features. Unlike our method, and similar to SuGaR [6], this method does not produce the necessary standardised mesh topology, making it unable to optimise Gaussians on footage of a moving subject, which is necessary for high-quality animated Gaussian head avatars. We will overcome this issue by using a 3DMFM [4].

SplattingAvatar [28] expresses a Gaussian's position as (k, u, v, d), where (u, v) are barycentric coordinates local to its parent triangle k, and d is the displacement along the normal. During optimisation, when a Gaussian's (u, v) coordinate crosses the triangle boundary, it gets bound to its neighbouring triangle. Allowing the Gaussians to "walk" over the mesh during optimisation ensures that they are always bound to the triangle directly below them. In contrast, our method restricts Gaussians to change parents, which allows them to better model often occluded surfaces such as the inside of the mouth.

GaussianHair [17] models hair as strands of thin, elongated Gaussians connected in series. To do this, the authors first use an off-the-shelf decoder to extract coarse hair strands from their ground-truth. Then, Gaussians are optimised onto those strands. This approach gives all the benefits of efficient high fidelity rendering that the Gaussians provide, with the added benefit that the Gaussians can be animated using conventional hair simulations. In contrast to the above-mentioned strand representation, which specialises in modelling hair, our method binds Gaussians to a triangle mesh. This more general approach allows us to model faces and hair.

Unlike the methods discussed above, our method addresses the inherent issue of using 3D Gaussians to represent opaque surfaces by obscuring Gaussians which should not be visible if the surface of the mesh were truly opaque.

3.2. Editing Gaussian Scenes

An artist should be able to modify a 3D scene to match their intended vision. This also holds for scenes using 3D Gaussians. This section explores prior work in editing 3DGS scenes.

Deepfake for the Good [29] creates edited Gaussian head avatars by modifying the ground-truth footage prior to the optimisation process using a deepfake face-swap. In this way, they combine the characteristics of different faces. Edits are applied before the optimisation process, which means that an avatar needs to go through the expensive optimisation process again before seeing the result of the edit. In contrast, our method edits avatars after they have been optimised. This allows our method to be interactive and display the result of the edit in real time.

GaussianEditor [3] uses a text-based editing approach for Gaussian scenes. A 2D diffusion model is used to guide the updates to the scene according to a text prompt. To ensure accurate editing of the Gaussians to be updated, a semantic tracing tag is added to each Gaussian, enabling precise segmentation. Additionally, the editor supports the removal and addition of objects given a text prompt and a 2D mask from a specific camera viewpoint. Like our method, this editor is interactive, but it does not support the editing of mesh-bound Gaussians, which are needed for animation, nor does it provide dedicated tools for editing faces.

SplatShop [27] is an editor for large-scale Gaussian scenes with a Photoshop-like approach to editing. Scenes are edited by selecting Gaussians and modifying them. Selections can be deleted, cloned, translated, scaled, and rotated. A paintbrush tool allows users to paint on top of the existing Gaussians, and an asset library lets them add more models to the scene. Impressively, the editor supports all of these operations, including the ability to undo/redo certain operations, all while running in VR. Like GaussianEditor [3], this editor also does not support editing mesh-bound Gaussians, nor does it provide tools to edit Gaussians at the semantic level.

While the work discussed above focuses on the editing of unbound Gaussians, our method specifically addresses the need for editing mesh-bound Gaussians by contributing a novel approach based on semantic mesh segmentation.

4

Methodology

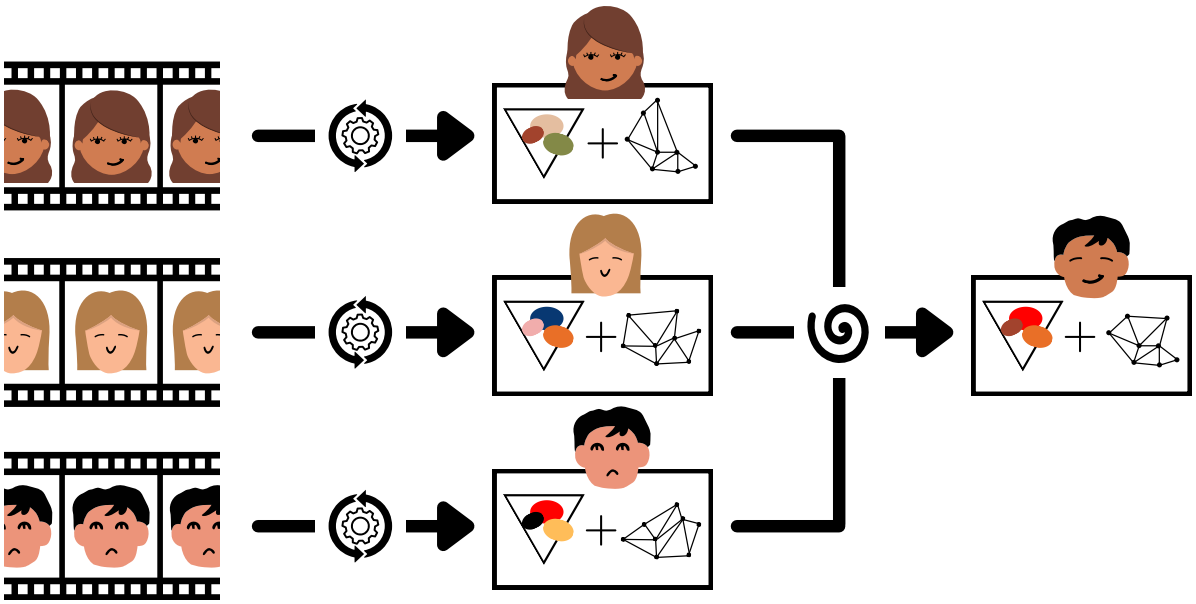


Figure 4.1: An illustrated pipeline of our method for editing Gaussian head avatars. Video footage of human faces is used to generate a database of source avatars. Each avatar is composed of a set of mesh-bound Gaussians and an underlying mesh. A composite avatar is created by selectively combining source Gaussians and mesh blending. The selection is done according to a semantic segmentation of the underlying mesh. The composite avatar in the illustration is created from the shape and aesthetic features of the three source avatars as indicated by the eyes, mouth, face shape, and skin colour.

We address the inability to edit animatable Gaussian head avatars by introducing a method for shape and texture transfer of Gaussian head avatars. Our method of editing works at the semantic level based on mesh segmentation, allowing the user to selectively combine features from a database of source avatars to create a composite avatar.

We also present 3 additional improvements, which are crucial for the quality of a composite avatar. Specifically, we adapt Qian et al.'s [21] Gaussian optimisation method to better represent an opaque surface, add a post-processing step to smooth the Gaussians between semantic regions, and add a colour correction step to better match skin colour between source models.

To demonstrate our method, we built an interactive editor on top of GaussianAvatars [21] for its realistic, animatable Gaussian head avatars.

4.1. Novel Face Synthesis

We define a source avatar $f = (\mathcal{G}, \mathcal{T}, V)$ with mesh-bound Gaussians $\mathcal{G}$, mesh topology $\mathcal{T} \subset \{1, 2, \ldots, n\}^3$ and mesh vertex positions $V = \{v_1, v_2, \ldots, v_n\} \subset \mathbb{R}^3$. The vertices of a triangle $t = (i, j, k) \in \mathcal{T}$ are defined by (v_i, v_j, v_k). A composite avatar $f' = (\mathcal{G}', \mathcal{T}, V')$ is then created from a set of source avatars $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ with identical mesh topology $\mathcal{T}$. To edit f', we partition $\mathcal{T}$ into a set of semantic regions $\mathcal{R} = \{r_{nose}, r_{mouth}, \ldots\}$. When talking about Gaussians, we use superscript to denote the source avatar and subscript to denote its binding. That is, g_r^f refers to the Gaussians of the source avatar f bound to the region r.

4.1.1. Combining Gaussians From Different Avatars

We create $\mathcal{G}'$ by combining subsets of source avatars' Gaussians $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m$. In practice, we do this per region by assigning a source avatar to each region of the composite avatar. In this way, a composite avatar's Gaussians are a patchwork of the Gaussians of its source avatars.

Note that all the avatars have the same mesh topology, allowing us to transfer Gaussians from one avatar to another while preserving their original binding and semantic meaning.

4.1.2. Combining Face Shapes From Different Avatars

Next, we describe how to combine face shapes from multiple avatars to create a new composite mesh. To do that, we create V' as a per-region convex combination of source avatars' vertex positions $V_1, V_2, \ldots, V_m$ according to a slightly modified version of the technique described by Ma et al. [18]. They assign weights to control points in each region and then compute the weight of each vertex by interpolating the k nearest control points, resulting in a smoother transition between regions. We have the benefit that our mesh does not need to be very smooth since the Gaussians mask any boundary artefacts. This allows us to use the same weight across an entire region. Additionally, our regions are defined by triangles and not vertices, so we assign a weight to each triangle edge instead of each vertex.

Per region r, we assign to each source avatar f a weight $w_r^f \in [0, 1]$ such that $\sum_f w_r^f = 1$. We use these weights to calculate the convex combination of the directed edges e_r in each region r as follows:

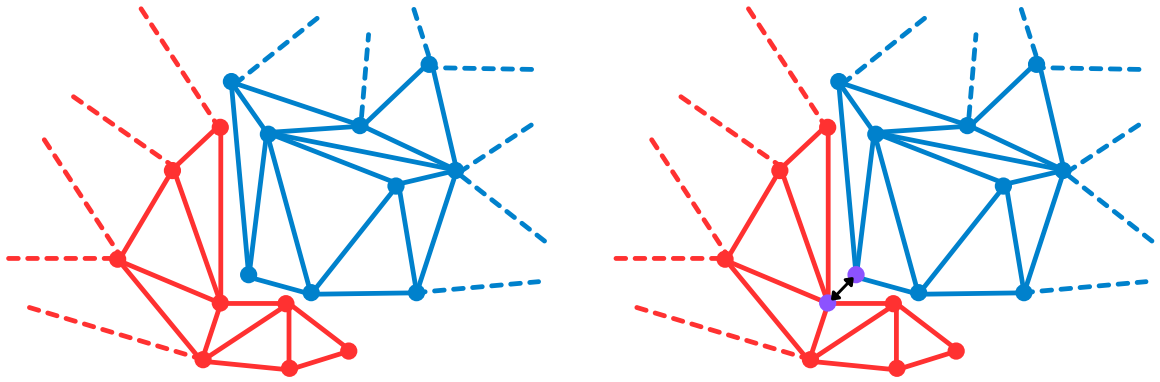
$$d_r = \sum_{f=1}^{m} w_r^f \cdot e_r \quad \text{where} \sum_{f=1}^{m} w_r^f = 1$$

We create a $3 \cdot |\mathcal{T}| \times n$ oriented incidence matrix M where each row represents a directed edge $e = (i, j)$ of a triangle in $\mathcal{T}$ and each column represents a vertex v.

$$M_{e,v} = \begin{cases} -1 & \text{if } v = i \\ 1 & \text{if } v = j \\ 0 & \text{otherwise} \end{cases}$$

Then, to calculate V', we solve the linear system $MV' = d$. As shown by Ma et al. [18], we can multiply both sides of the equation by M^T to get $M^T MV' = LV' = M^T d$ where L is the Laplacian matrix, which is efficiently solved using a pre-computed LU decomposition.

One caveat of this approach is that because there are no edges between disjoint sections of the mesh, the solved V'does not preserve their relative placement. This is an issue, specifically because face meshes tend to use disjoint meshes for the eyeballs and the teeth. To overcome this, we add artificial edges between chosen anchor points on the disjoint meshes. Given anchor points a_d on a disjoint part of the mesh and a_m on the main mesh, we add edges (a_d, a_m) and (a_m, a_d) to M (see Figure 4.2). To assign a weight to the edge, we consider it in the region of its starting vertex. In practice, we make each disjoint mesh its own region in $\mathcal{R}$ and take the geometric median of a region as its anchor point. Then we connect each disjoint part to 2 nearby regions.



(a) Two disjoint meshes which are part of a singular mesh.

(b) Adding directed edges to anchor points on two disjoint meshes.

Figure 4.2: Illustration indicating the addition of two edges to artificially create a single joint mesh for shape blending. The disjoint meshes are indicated by blue and red. The extra edges are black, and the anchor points are purple.

Lastly, because our matrix does not have full rank, we need to pin some vertex p to some position u when solving for the vertex positions. This ensures the mesh does not get translated relative to its previous position. Intuitively, this makes sense if you consider that we have only placed constraints on the edges of the mesh but not on the vertex positions, meaning a mesh can be translated while still fulfilling the constraints on the edges. For p we take the geometric median of the mesh, and for u we take the position of v_p in V_0. To add this constraint to our system, we modify L and $M^T d$ such that:

$$L_{p,i} \leftarrow \begin{cases} 1 & \text{if } i = p \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad (M^T d)_p \leftarrow u$$

4.2. Artefact Reduction in Composite Avatars

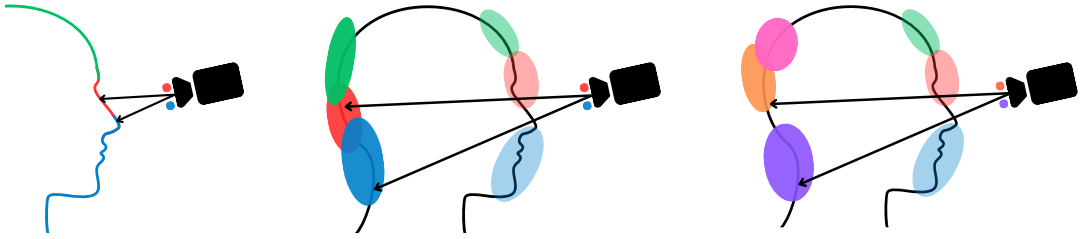
The approach described so far in the previous section suffers from artefacts stemming from the combination of Gaussians from multiple avatars. For this reason, we provide several improvements in the quality of composite avatars. These include a method to reduce interference caused by the transparency of Gaussians (see Section 4.2.1), and two post-processing methods to improve the homogeneity of the Gaussian layer (see Section 4.2.2).

4.2.1. Mitigation of Misleading Transparency Effects

When rendering an opaque surface (e.g. a face), occluded surfaces must be excluded from the render. Unfortunately, as a rendering technique, Gaussian splatting specifically includes all Gaussians in the render until the saturation of a ray. In mesh-bound Gaussians, this results in interference from Gaussians representing occluded surfaces. For example, during optimisation, Gaussians on the other side of the mesh can constructively interfere such that from the point of view of the camera, the scene matches the training footage, but the Gaussians don't actually form an opaque surface (see Figure 4.3b).

Unfortunately, in a composite avatar, the effect is amplified because Gaussians that were causing constructive interference might have been swapped out, resulting in misleading results as seen in Figure 4.3c.

This can be mitigated by using training footage with 360° views will mitigate the problem, because then there is footage from all sides of the head to inform the Gaussians on all sides what they *should* look like. Unfortunately, many face capture scenarios, including the popular NeRSemble dataset [13] used in this thesis, only have front-facing camera angles. To address the issue of interfering Gaussians, we introduce two simple modifications to the optimisation method of Kerbl et al. [12].

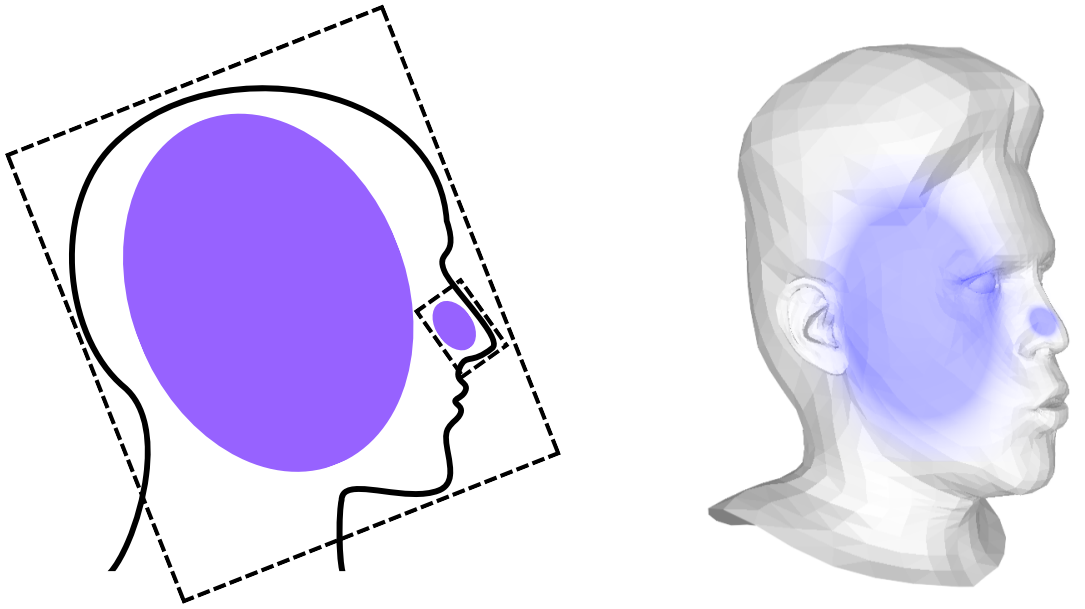


(a) Cross-section of ground-truth. **(b)** Cross-section of source avatar. **(c)** Cross-section of composite avatar.

Figure 4.3: Diagrams illustrating the effect of holes on composite head avatars. The dot next to a ray indicates the colour of its pixel. Note how, despite including the face Gaussians of the source avatar in (b), the rays of the composite avatar in (c) will not render as the intended red and blue from the ground truth (a).

Obscuring Gaussians

Our goal is to augment the optimisation, such that we obscure Gaussians bound to occluded surfaces, preventing them from causing interference. We achieve this by artificially adding opaque Gaussians inside the head as seen in Figure 4.4. Because they are fully opaque, all Gaussians behind them will get obscured. Ideally, the obscuring Gaussians should fill the convex hull of the mesh. Because our mesh is different for each frame of footage, we need an efficient solution to determine the placement of obscuring Gaussians. Fortunately, we can achieve a decent approximation using a very simple heuristic. We take advantage of the fact that the shape of the human head vaguely approximates an ellipsoid, allowing us to fill it by simply placing one large Gaussian in its centre. The rotation and scale of our Gaussian are determined by an oriented bounding box (OBB) of the head. The rotation of the Gaussian is simply the rotation of the OBB. The scale is OBB's extents scaled down by a constant factor to ensure that the Gaussian doesn't bleed through. The scaling factor is empirically chosen to be 0.09.



(a) Obscuring Gaussians placed according to the OBB indicated by the dotted line. **(b)** Obscuring Gaussians placed inside a mesh.

Figure 4.4: Visualisation to illustrate the placement of obscuring Gaussians inside the head of an avatar. This will prevent Gaussians on the other side from being visible to the renderer during optimisation.

We do not bind these Gaussians to any triangle, nor optimise their parameters. To prevent Gaussians from adapting to the obscuring Gaussians, we change their colour every iteration. In practice, we cycle through red, green, and blue to simulate random noise. This simple addition encourages the Gaussians to optimise into a water-tight opaque layer.

Back-Face Triangle Culling

The optimisation method of Qian et al. [21] requires that every triangle have at least one Gaussian bound to it. This forces Gaussians to be bound to triangles even if there is no training data in which that part of the head would be seen. For example, when using only front-facing footage, the back of the head is not visible; however, Gaussians will still be bound there.

Instead, we modify the mesh topology prior to optimising Gaussians by culling all triangles which lack sufficient ground-truth coverage. Given a camera c with view direction $\vec{c}$, we consider a triangle t with normal $\vec{n}_t$ visible if it is facing towards the camera. In other words:

$$v(c,t) = \begin{cases} 1 & \text{if } \vec{n}_t \cdot \vec{c} < 0 \\ 0 & \text{otherwise} \end{cases}$$

In this way, we sum a triangle's total view count across every training camera and expression. We divide the view count by the number of cameras and expressions to get the view ratio. Finally, all triangles with a view ratio less than a threshold τ are culled.

$$\frac{\sum_{c=1}^{n_c} \sum_{x=1}^{n_x} v(c, t_x)}{n_c n_x} < \tau$$

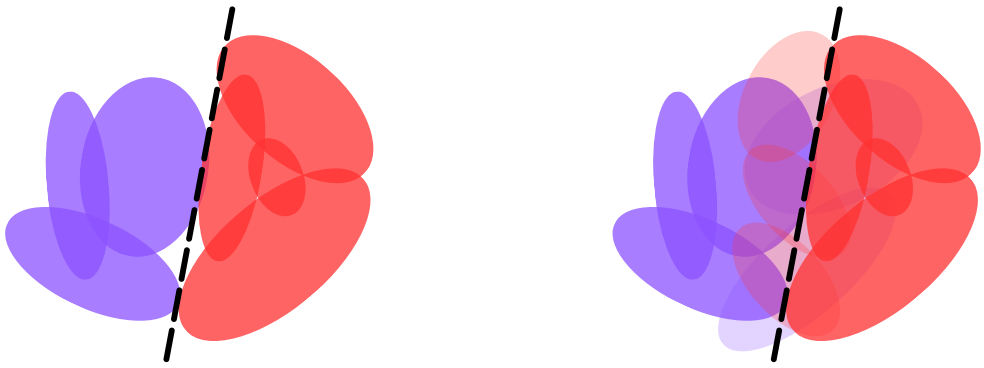
Where t_x refers to the orientation of triangle t at expression x. The number of cameras and expressions is given by n_c and n_x, respectively.

4.2.2. Post-Processing of Gaussians

We introduce 2 post-processing techniques to homogenise G'. The first technique smoothens the borders between regions, and the second targets differences in skin tone.

Border Smoothing

The borders between two regions with different source Gaussians can appear quite jarring. The transition between regions is smoothed by simulating alpha blending (see Figure 4.5).



(a) Without border smoothing.

(b) With border smoothing.

Figure 4.5: An illustrated example of the border smoothing method. The border between the regions is represented by the dotted line. Colour indicates the source avatar.

We extend $\mathcal{G}'$ with additional Gaussians on the boundary triangles between these regions. A triangle t borders region r iff $t \notin r$ and t shares a vertex with any triangle in r. The function $b(r,p)$ returns all triangles in region p which border on region r. The set of extra Gaussians is given by:

$$\bigcup_{r,p \in \mathcal{R}} g^{f_r}_{b(r,p)} \quad \text{where } f_r \neq f_p$$

To avoid adding unnecessary Gaussians, we only include Gaussians for bordering regions with Gaussians from differing sources. Lastly, we halve the opacity of the extra Gaussians to simulate alpha blending on the border.

(a) Avatar with discoloured eye region.

(b) Avatar with discoloured nose and mouth regions.

Figure 4.6: Composite avatars with visible discolouration as a result of combining source avatars with different skin tones.

Skin Tone-Based Colour Correction

Combining source avatars with different skin tones typically produces lacklustre avatars, as shown in Figure 4.6. For this reason, we introduce a colour correction technique which modifies the colour of Gaussians in one region to better match the colour of a specified base avatar.

For this purpose, we apply the colour correction per region by calculating a weighted mean $\bar{c}_f$, where f is the source avatar assigned to the region. We use $w_g = \alpha_g \cdot \text{size}(s_g)$ as the weight for Gaussian g, where $\text{size}(s) = \frac{\prod s}{\min(s)}$ approximates the largest cross section of a Gaussian. For a Gaussian g in a region, we compute the corrected colour c'_g according to:

$$c'_g = c_g \cdot \frac{\bar{c}_{base}}{\bar{c}_f} \quad \text{with} \quad \bar{c}_f = \frac{\sum_g c_g w_g}{\sum_g w_g}$$

5

Implementation

Our implementation is built on top of the existing GaussianAvatars [21] codebase, which is written primarily in Python 3 using NumPy, and PyTorch [20] with CUDA. Gaussians are rendered using the original renderer by Kerbl et al. [12]. Meshes are rendered with Nvdiffrast [14]. The user interface (see Figure 5.1) is built with DearPyGUI [8]. We use SciPy [32] and SuperLU [16] to solve for vertex positions when blending face shapes. The object-oriented bounding box for the obscuring Gaussians is computed with Open3D [34].

Figure 5.1: A screenshot of our editor. The bar on top displays a preview of the source avatars. The top left shows the debug panel, which can be used to toggle debug options such as changing the scale of the Gaussians or displaying the underlying mesh. The panel in the bottom left is used to edit the avatar by assigning Gaussians and modifying shape coefficients per region, as well as toggling the post-processing methods on/off. The bottom right panel displays information about the Gaussians and the mesh. Lastly, the panel in the top right is used to animate the avatar by selecting a set of expression parameters and iterating through them. It is also possible to manually adjust the joint parameters and the first 5 expression parameters.

For our source avatars, we use GaussianAvatars [21], consisting of fine-tuned FLAME parameters and mesh-bound Gaussians. These are trained using the same optimiser, learning rates, and parameters as reported by Qian et al. [21].

Our hole-filling methods are specifically implemented to not interact with the optimiser's computation graph. Obscuring Gaussians are not bound to any triangle and are passed only to the renderer and not tracked by the optimiser. Back-face triangle culling is applied before training starts. Because this changes the mesh topology, we map the Gaussian bindings back to the original topology when exporting.

Similar to the work by Qian et al. [21], we generate the underlying mesh at runtime using a pre-trained FLAME 2023 model [15]. To animate a composite avatar, we first recompute the meshes of its source avatars according to the next frame's expression parameters. Then, using the same mesh blend coefficients, we recalculate the composite avatar's vertex positions.

To handle the manipulation of multiple Gaussian datasets, we load all the source Gaussians into one large unified buffer. This lets us efficiently select which Gaussians to render using an index mask. Every time the regional assignment changes, we rebuild the mask using pre-computed lookup tables. We also use lookup tables for the boundary Gaussians and the mean colour to speed up post-processing.

For editing, we use 18 regions grouped into 7 distinct groups (see Figure 5.2). The mesh regions are based on Qian et al.'s [21] modified version of the FLAME [15] vertex masks. Our changes make sure every triangle belongs to exactly one region by removing overlapping triangles and adding *left_eye_socket*, *right_eye_socket*, and *remainder* regions to cover unassigned triangles. The *eyeball* and *teeth* regions are disjoint from the rest of the main mesh. We connect each *eyeball* to its respective *eye_socket* and *eye_region* and each *teeth* region to the *lips* and *lips_inside* regions.

Group	**Regions**		**Triangles**
Mouth	*teeth_upper* *teeth_lower*	*lips* *lips_inside*	740
Eyes	*left_eye_region* *left_eye_socket* *left_eyeball*	*right_eye_region* *right_eye_socket* *right_eyeball*	3495
Ears	*left_ear*	*right_ear*	2306
Neck	*neck*	*boundary*	566
Hair	*hair*	*remainder*	776
Nose	*nose*		718
Face	*skin*		1543

Figure 5.2: The semantic groups and their respective sub-regions used in our editor. Edits are applied universally across every region in a group.

6

Evaluation

To demonstrate the potential of our approach, we provide a qualitative analysis of our method (see Section 6.1) and a quantitative analysis (see Section 6.3) based on a user study investigating the perceived authenticity of source and composite avatars (see Section 6.2). Lastly, we examine the real-time performance of our editor (see Section 6.4) to evaluate the interactivity of our method.

6.1. Qualitative Analysis

In this section, we will visually explore the effect of our hole-filling, border smoothing, and colour correction methods on composite avatars. We do this by selecting 4 composite avatars, which we think display the strengths and limitations of our method. We show 4 versions of each composite avatar. A version without any of our improvements, one with only hole-filling, one with hole-filling and border smoothing, and one with all three operations applied. The composite avatars and the subjects they are based on are presented in Table 6.1.

The impact of hole-filling is generally negligible on a still image but becomes more prominent on a moving avatar. The sequence of images in Table 6.2 better illustrates the issue of transparent regions of the face, and how it is resolved by our method. One negative consequence of our hole-filling method is that it increases the opacity of Gaussians, which makes borders between Gaussians from different sources more distinct. This becomes apparent when looking at the mouth and eyes of avatar C and the nose and eyes of avatar D (see Table 6.1).

Our border smoothing method addresses this issue by smoothing out the increased harsh borders. This is visible at the eyes of avatar D, the neck of avatar B, and the nose of avatar D (see Table 6.1). Another benefit of the border smoothing method is that it covers up holes where the Gaussians do not fully cover a border, such as the forehead of avatar B (see Table 6.1).

Differences in skin tone between subjects are tackled by our colour correction method. Its effect is visible at the neck of avatar A, the eyes and mouth of C, and the eyes, nose, and mouth of D (see Table 6.1). Besides just the skin tone, it can also match the hair colour, as seen in avatars A and B (see Table 6.1).

Our method for colour correction does not distinguish between skin and non-skin Gaussians, such as hair or clothing. This can be beneficial such as you see in avatar A's hair, which better matches the hair colour of the base subject (see Table 6.1). On the other hand, it can be detrimental, such as you see in the neck of avatar B, which has gained a greenish tint due to trying to match the colour of the base subject's shirt, or in the shirt of A, which has darkened to match the colour of the base subject's shirt (see Table 6.1). A simple workaround can be to simply take the source avatar assigned to the neck as the base colour, ensuring that colour correction is never applied to the neck region, avoiding discolouration caused by clothing.

	Base	Others	None	Fill	F+Smooth	All
A						
B						
C						
D						

Table 6.1: Examples of composite avatars showcasing the performance of our hole-filling, border smoothing, and colour correction methods. The left 2-3 images are of the human subjects used to create the composite avatars. The remaining images show the avatar without any of our improvements applied, with only the hole-filling operation applied, with both the hole-filling and border smoothing operations applied, and with all three operations applied.

No Fill			
No Fill + Hair			
Fill + Hair			

Table 6.2: Visual comparison of our hole-filling method. Each row shows an avatar under rotating camera views. The first row demonstrates how the back-side Gaussians hide the transparency in the face. The second row makes this clear by changing the hair Gaussians (and thus the back-side Gaussians), while the last row shows how our hole-filling method resolves the issue.

Subject	264	460	460 + CC
264			
460			

Table 6.3: Visual comparison of the impact of shape blending compared to modifying the Gaussians. The composite avatar is composed of subjects 264 and 460 (see Appendix A), with the former as the base. Modifications are only applied to the nose region, and shape coefficients are all-or-nothing, i.e. 100% for one source and 0% for the other. The row indicates which source's shape is used. The columns indicate which source's Gaussians are used. Colour correction is applied in the last column. Note how even when the Gaussians have not been modified, the result of the shape modification (a smaller nose) is easy to overlook and even more so when the Gaussians are modified.

6.2. User Study

To investigate the perceived authenticity of source and composite avatars, we conducted a user study with 22 participants. The 22 participants were each shown a mix of source and composite face avatars and surveyed on whether they thought the avatar they were shown was edited or not. For the avatars identified as edits, we further asked them to select which parts they think were modified.

6.2.1. Creation of Test Avatars

For the study, we created a database of 14 source avatars from pre-processed NeRSemble [13] data provided by Qian et al. [21]. The subjects and expressions are listed in Table 6.4 with images in Appendix A. Each source avatar is synthesised from 11 video sequences of a subject performing various facial expressions. Each sequence is captured from 16 distinct front-facing viewing angles.

Expressions						
EMO	1	2	3	4		
EXP	2	3	4	5	8	9
FREE						

Subjects					
074	104	140	165	175	210
218	238	253	264	302	304
306	460				

Table 6.4: The 11 sequences and 14 subjects from the NeRSemble [13] dataset which the source avatars are trained on.

Composite Avatars

Composite avatars are randomly created using 3 source avatars from the database. We designate 1 avatar as the base and apply random modifications to the coefficients and Gaussian assignments according to the following rules:

- Edits are applied as a whole to every region in a group. We use the following groups: eyes, nose, mouth, hair + neck + ears (HNE).
- The Gaussians of at least one group are replaced with those of an additional avatar.
- The shape of at least one group is blended with the shapes of the additional source avatars. The coefficients are randomly decided using a stick-breaking approach.
- The shape of the HNE group is not blended and is instead matched to the shape of the source avatar assigned to it.

Hair often overlaps the ears and the neck, leading to poor results when swapping those Gaussians separately. Additionally, if Gaussians are attached to a drastically different mesh than the one they were optimised on, the effect breaks down. This tends to be the case with hair, because the difference in hair shape between avatars is substantial. These are known issues with our method, so to avoid them biasing the result, we have grouped HNE together and additionally matched its shape to that of its source avatar by restricting the blend coefficients for that region.

6.2.2. Categorisation of Test Avatars

We compare our composite avatars to unedited source avatars to show that the realistic quality of the avatars is preserved. We evaluate our hole-filling improvement by comparing the performance of hole-filled source avatars to non-hole-filled source avatars, and the performance of hole-filled composite avatars to non-hole-filled composite avatars. Lastly, we evaluate our border smoothing improvement by comparing the performance of hole-filled composite avatars with and without the border smoothing applied. The subtle impact of shape blending is overshadowed by edits to the Gaussians (see Table 6.3), and is not directly evaluated in our study to maintain focus. As mentioned earlier, our colour correction technique is excluded from this study. Table 6.5 contains a summary of the 5 test categories used in the study.

Category	Description
edit_original	Composite avatars created using non-hole-filled source avatars.
edit_filling	Composite avatars created using our hole-filled source avatars.
edit_filling_smooth	Composite avatars created using our hole-filled source avatars with our border smoothing technique applied.
unedited_original	Source avatars optimised using GaussianAvatars' original method
unedited_filling	Source avatars optimised using our hole-filling method.

Table 6.5: An overview of the test categories in our study

In the study, we used 45 randomly created composite avatars (15 for each edited category) and 14 source avatars (7 for each unedited category) for a total of 59 avatars ordered randomly for each participant. The subjects of both unedited categories were kept distinct to prevent familiarisation bias. The generated avatars for the survey are listed in Appendix B.

6.2.3. The Testing Interface

The participants performed the task in a custom-built interactive graphical user interface running on a computer(see Figure 6.1). The window is locked to a resolution of 800x1200. The participants were able to use a computer mouse to change the camera view used to observe the avatar according to their preference. However, the camera was locked to orbit the avatar at a radius of 2.5 world-space units with ±45° horizontal and ±15° vertical freedom.

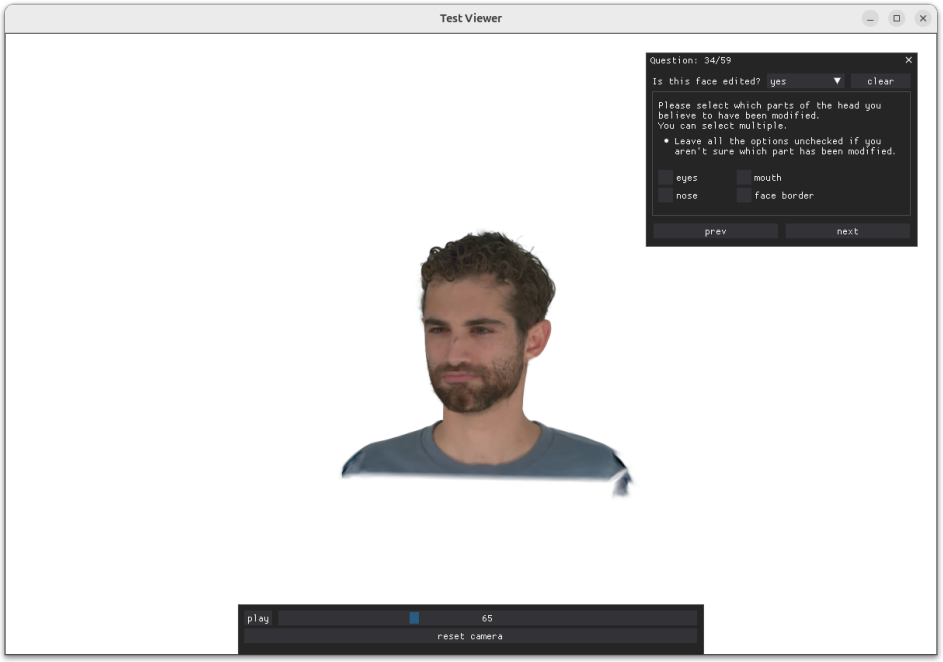


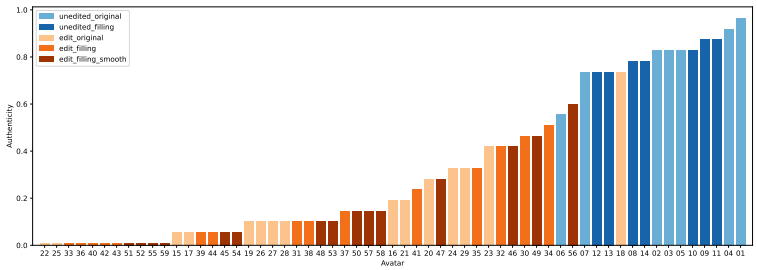
Figure 6.1: The testing interface used in our study.

The avatars were all animated with the same looping sequence of closed-mouth expression parameters sourced from subject 306. The animation sequence was capped at 25 FPS, matching the downsampled frame rate at which the avatars were trained. Because open-mouth expressions usually have a lot of artefacting in and around the mouth, we exclude them from our study.

The study was conducted on an Alienware Area-51 R5 computer with an Intel Core i7-9800X CPU, 32GB of RAM, an RTX 4070 with 12GB of VRAM, running the Ubuntu 24.04 LTS operating system.

6.3. Quantitative Analysis

To analyse the authenticity of an avatar, we calculate a score based on the mean response to the question "Is this face edited?". We define this authenticity score as the ratio of "no" responses to the total number of responses per avatar. This score represents the consensus of participants regarding the authenticity of a specific avatar, where a higher number indicates more participants considered this avatar unedited. An in-depth overview of the results per test case is listed in Appendix B. An overview of the authenticity scores for each avatar is shown in Figure 6.2.



Category	Mean
unedited_original	0.80
unedited_filling	0.79
edit_original	0.19
edit_filling	0.15
edit_filling_smooth	0.16

Figure 6.2: Overview of the authenticity scores from our survey. Shades of blue indicate the two categories of unedited avatars, and different shades of orange indicate the three categories of edited avatars.

Comparing mean authenticity scores (see Figure 6.2), we see that both categories of unedited avatars are perceived as similarly authentic. Additionally, on average, the edited avatars are perceived as significantly less authentic than the unedited avatars. Out of the edited categories, we see that *edit_original*, without our improvements, is perceived as the most authentic on average. The categories using our improvements score equivalently poorly on average.

Despite the negative results implied by the mean authenticity scores, a plot of individual authenticity scores shows us that edited avatars 56 and 18 outperform the unedited avatar 6, with unedited avatar 18 even tying with three more unedited avatars. This indicates that edited avatars can achieve comparable authenticity to unedited avatars. The 5 most authentic edited avatars, aside from avatar 18, use our improvements. However, from the 11 avatars which are considered 100% inauthentic, 9 use our improvements. This conflicting data shows us an inconclusive result, and in general, we see a very large spread between the authenticity scores spanning from 0 to 0.74 for edited avatars.

To get a better idea of why the spread is so large and why some edited avatars are perceived as so much more authentic than others, we compare the 3 most authentic (Table 6.6) and the 3 least authentic (see Table 6.7) avatars. What stands out is that, unlike the authentic avatars, the inauthentic avatars are typically composed of conflicting features. For example, facial hair on a female face (1X, 3X, 3Y), a masculine face with a feminine neck or hairstyle and vice versa (1X, 1Y, 2X), or clashing skin tones (2X, 2Y, 2Z) (see Table 6.7). On the other hand, the authentic avatars are very homogeneous in their composition.

Category		A	B	C
edit_original	1			
edit_filling	2			
edit_filling_smooth	3			

Table 6.6: The 3 most authentic edited avatars in decreasing order of authenticity.

Category		X	Y	Z
edit_original	1			
edit_filling	2			
edit_filling_smooth	3			

Table 6.7: The 3 least authentic edited avatars in decreasing order of authenticity.

Another issue we take note of is the apparent lack of Gaussians around the nose of 3Z and the inclusion of extra Gaussians around the nose of 2X (see Table 6.7). Upon further investigation, we found that the Gaussians for the hole-filling version of subject 175 are inaccurately bound to the mesh. Consequently, they do not match the semantic regions defined by the mesh triangles. In particular, the Gaussians bound to the mouth region of the source avatar for subject 175 extend to the nose (see Table 6.8), producing prominent artefacts. This explains why 2X, using 175's mouth, has additional Gaussians around the nose, and why 3Z, using 175's face but 264's mouth, is missing Gaussians around the nose (see Table 6.7).

Subject	Mouth	Face	Nose + Eyes	Hair
175				
302				

Table 6.8: Gaussians of various semantic regions of the hole-filled source avatar of subject 175 compared to the more accurate bindings of subject 302. Note the excess Gaussians around the nose, which are bound to the mouth triangles. These should be bound to either the face or nose triangles, which now show some transparency around the area of the nose. Additionally, the hair Gaussians bound to the hair triangles contain Gaussians which are placed inside the head and neck of the mesh.

We conclude that the authenticity of a composite avatar is heavily dependent on the compatibility of the source avatars it is composed of. This has to do with the implausibility of certain combinations of facial features in real life, such as clashes in racial and gendered facial features. Due to the randomised nature of the avatars used in our study, many of these implausible combinations are created. This makes it difficult to see if an avatar performed badly due to our method or due to its composition.

6.3.1. Exclusion of Implausible Avatars

To allow a better comparison between the different methods, we exclude implausible combinations according to the criteria described below. Note that an avatar may fit multiple criteria for exclusion.

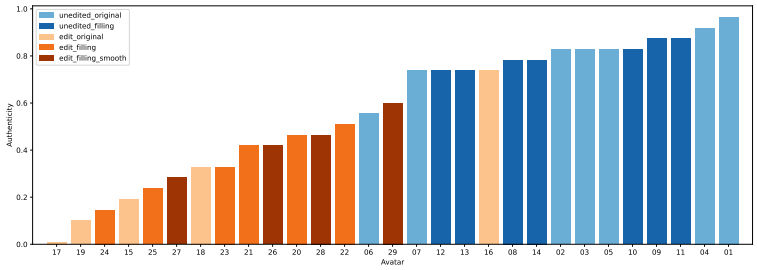
The hole-filled source avatar for subject 175 specifically has very inaccurate Gaussian bindings, making it unsuited for creating composite avatars (see Table 6.8). We exclude the 5 composite avatars using the hole-filled source avatar for subject 175.

The mouth region of male source avatars typically includes facial hair. This is a clear giveaway when combined with a female face. We exclude the 5 composite avatars with a male mouth and a female face.

Male necks tend to be bulkier, while female necks tend to be more slender. Hairstyles between men and women are typically quite different as well. Because the neck and hair are already grouped under NHE, we exclude the 17 composite avatars with conflicting genders for the face and NHE.

Lastly, a large difference in skin tone is also an obvious giveaway of an edit. Based on visual assessment, we categorise subjects 074, 175, and 264 (see Appendix A) as dark skinned and the remaining avatars as light-skinned. We exclude the 22 composite avatars with Gaussians from both skin tone categories.

In total, we exclude 30 of the 45 total composite avatars matching one or multiple of the criteria described above. A full overview of the excluded avatars, including the reason(s) for their exclusion, is provided in Appendix C.



Category	Mean
unedited_original	0.80
unedited_filling	0.79
edit_original	0.26
edit_filling	0.34
edit_filling_smooth	0.43

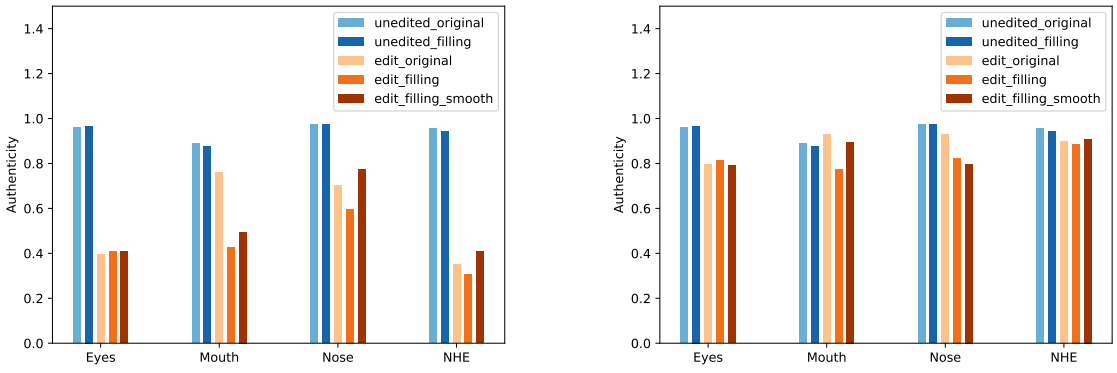
Figure 6.3: Overview of the authenticity scores from our survey after filtering out implausible avatars.

After filtering out implausible avatars, the *edit_filling* category has a score of 0.34 outperforming *edit_original*'s score of 0.26. This shows that our hole-filling method is an improvement when used for creating composite avatars while preserving the quality of unedited source avatars. We conclude that our hole-filling method is an improvement with respect to the GaussianAvatars approach [21]. Furthermore, *edit_filling_smooth* has a mean score of 0.43, outperforming the *edit_filling* category. We conclude that our border smoothing method further improves the perceived authenticity of composite avatars.

Despite the high quality of some of the most authentic avatars, the edited categories do not match the authenticity mean scores of the unedited categories. A limitation of our study is that the source avatars used to create the composite avatars are shown in the same survey, making it possible to extrapolate which avatars are edited by identifying the source models. We conclude that the participants can accurately distinguish source avatars from composite avatars.

6.3.2. Analysis of Regional Authenticity Scores

Using the same approach, we can determine the authenticity scores for each facial region (see Figure 6.4). In this way, we investigate if there are specific regions which are more likely to be seen as inauthentic.



(a) Regional authenticity score for edited regions. Unedited categories are shown for comparison, but naturally have no edited regions.

(b) Regional authenticity score for unedited regions.

Figure 6.4: The authenticity score for each region used in our study, grouped by category.

The graphs in Figure 6.4 show again that the unedited avatars are generally perceived as authentic. The mouth region on unedited avatars is perceived as marginally less authentic. This could be related to the expression transfer used to animate the avatars, as it can cause artefacting, especially around the mouth. Looking at Figure 6.4b, we see that unedited regions on composite avatars, in general, have per-region authenticity scores which are a bit below those of the fully unedited source avatars. This could be explained by other edits to the avatar, lowering the overall perceived authenticity, resulting in participants falsely identifying unedited regions as edited.

Figure 6.4a indicates that the regions most likely to be perceived as inauthentic when edited are the

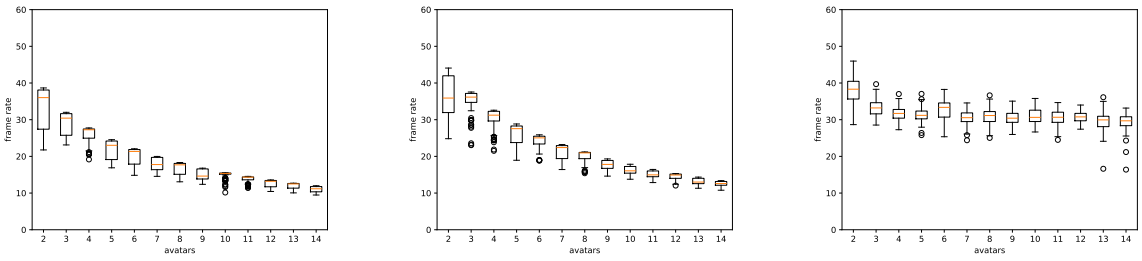
eyes and NHE, while the nose is more likely to be seen as authentic. However, these results must be taken with a grain of salt as the exclusion criteria defined in Section 6.3.1 are not applied, and so the results might be skewed by those particularly unnatural edits. We choose not to apply the exclusion criteria here because it reduces the data significantly (e.g. 1-2 avatars per region per category), making a meaningful analysis impossible.

6.4. Real-Time Performance

To evaluate the real-time performance of our editor, we performed some rudimentary benchmarks. We examine the frame rate of our editor while animating the avatar, modifying the face shape coefficients, and modifying the Gaussians assigned to each face region. To compare the effect the number of source avatars has on the frame rate, we run each benchmark once for 2 to 14 source avatars.

Each benchmark is run for 110 iterations, of which the first 10 are warm-up iterations. For the animation benchmarks, a sequence of 110 FLAME expression parameters, taken from one of the source avatars, is used. The shape coefficients and the Gaussian assignments are randomly determined. During each iteration, we disable the Python garbage collector to get clearer results.

The benchmarks were run on an Alienware Area-51 R5 computer with an Intel Core i7-9800X CPU, 32GB of RAM, an RTX 4070 with 12GB of VRAM, running the Ubuntu 24.04 LTS operating system.



(a) Expression animation. **(b)** Modifying shape coefficients. **(c)** Modifying Gaussian assignment.

Figure 6.5: Benchmark results over 100 iterations for 2 - 14 source avatars.

6.4.1. Results

The benchmark results (see Figure 6.5) show that our editor performs each action at ≥ 15 FPS for up to 7 avatars. Using more than 7 avatars is unrealistic, as there are only 7 semantic groups to edit. Furthermore, this benchmark measures the unrealistic worst case in which the editor performs a costly operation every frame. We consider the performance of our editor sufficient for real-time editing.

Comparing the results in Figures 6.5a and 6.5b to those in Figure 6.5c shows that the former scales poorly with the number of source avatars. This is a result of the way mesh deformation is implemented in the editor. The calculation of the composite mesh vertices is relatively expensive, and it is recomputed for every change to either the source meshes or the shape coefficients, which scales with the number of source meshes. Modifying the Gaussian assignments is achieved via index masks, which are precomputed and stored in lookup tables, making them scale very well.

7 Discussion

This chapter discusses the strengths and limitations of our method and implementation, as well as future work which can be done to improve our method.

7.1. Strengths

Despite some restrictions, composite avatars can be created using features from visually distinct subjects, creating unique but authentic avatars as shown by our results. Our method can generate suitable source avatars using front-facing-only footage due to our use of obscuring Gaussians. We also support the combination of source avatars with differences in skin tone thanks to our colour correction method.

Due to the conceptually simple, semantic-based approach to editing used by our method, we are able to create an editor which is more intuitive and accessible than more complex and technically demanding 3D editors. Additionally, because our method relies on the direct manipulation of Gaussian and triangle primitives, edits can be applied interactively, as shown by our benchmark results showing the real-time performance of our editor.

7.2. Limitations

A Gaussian's parent triangle determines which semantic region it belongs to. This is a decent approximation, but it allows for Gaussians to be bound to one region while being spatially in another. This is most prominently seen in Figure 6.8. In general, Gaussians for one region could include Gaussians for another region, e.g. the nose region containing a cheek Gaussian and vice versa. This becomes particularly problematic when the regions are assigned different source avatars.

The mesh used in our implementation does not include a surface for the mouth cavity, only teeth. This means that the mouth cavity lacks proper triangles for Gaussians to bind to. Consequently, Gaussians depicting the inside of the mouth often end up bound to non-mouth regions. This is also the reason we excluded open-mouth expressions from our study.

The subjects from the NeRSemble dataset [13] used to synthesise our source avatars have their hair down, partially obscuring parts of their head. This interferes with the segmentation of the Gaussians, especially around the ears and neck. Preventing us from treating these as distinct areas in our study.

In general, our approach doesn't support hair very well. We use a mesh to represent the underlying structure of an avatar's hair. However, a mesh surface cannot accurately depict the complex and dynamic movement of human hair.

Lastly, the randomised nature of the composite avatars used in our study produced inconclusive results, making it difficult to accurately evaluate the impact of our improvements.

7.3. Future Work

As mentioned in Section 7.2, the strategy used to segment Gaussians into regions fails when Gaussians are positioned too far from their parent triangle. A more suitable binding strategy could be to use barycentric coordinates [28, 5] to guarantee spatial relevancy to the parent triangle. Alternatively, Gaussian regions could be completely decoupled from mesh regions by defining them independently of their parent triangles. This way, regions can be defined using a segmentation model (e.g. SAGA [2]), or simply by an artist manually selecting Gaussians for each region.

A more suitable underlying hair structure, such as strands [17], would allow the head avatars to support dynamic hair movement. A hybrid approach combining mesh-bound Gaussians for the surface of the face and Gaussian strands for the hair could work better.

Our simple heuristic approach of inserting obscuring Gaussians significantly improves surface coverage of the Gaussians. Like any heuristic, however, it is not perfect. In our case, the obscuring Gaussians only roughly fill the head. A more accurate space-filling method could theoretically perform better, but will likely significantly impact optimisation time. Additionally, the unexplained inaccuracy of Gaussian bindings in the hole-filled source avatar for subject 175 (see Figure 6.8) may be caused by the Gaussians optimising to "cover up" the obscuring Gaussian, indicating a potential flaw in the approach. A more pragmatic approach might be to remove Gaussians instead of obscuring them. This can be done by determining which Gaussians should be visible for each iteration of the optimisation, and only rendering and optimising the Gaussians visible in that iteration. Gaussian visibility could, for example, be determined based on triangle visibility.

Using a conventional triangular mesh for the underlying structure of the Gaussians works well if the mesh geometry does not deform too much. Unfortunately, this is not always possible when combining Gaussians from different sources. One of the reasons for this is that rescaling a 2D triangle cannot sufficiently inform the 3D scaling factor of a Gaussian. A more accurate approach could be achieved by binding Gaussians to a polyhedral mesh [5, 35].

The impact of our colour correction method for skin-tone homogenisation was not included in our study and should be analysed in the future. Additionally, our method makes no distinction between what is or is not skin and is applied to every Gaussian in a region. A more accurate result could be achieved with a more restrictive approach by distinguishing skin Gaussians from non-skin Gaussians, for example, by an artist or a segmentation model (e.g. SAGA [2]).

8

Conclusion

This thesis introduced an editor for composing animatable Gaussian head avatars by selectively combining the Gaussians and the intrinsic face shape of source avatars according to semantic facial regions. We illustrated the intuitive concept behind the editor and addressed some of the difficulties in composing Gaussian head avatars. Three techniques were proposed to tackle some of these difficulties, two of which were evaluated through an ablation study. To address the inherent flaws of using transparent Gaussians as opaque textures, we modified the GaussianAvatars approach [21] for optimising mesh-bound Gaussians. Our adaptation results in a 131% improvement in the perceived authenticity of composite avatars. Additionally, we introduce a post-processing technique to homogenise the Gaussians on the borders between semantic regions of a composite avatar by simulating alpha blending at the borders between semantic regions. We have shown that composite avatars created using both methods are perceived as 165% more authentic. Lastly, we showcased the effect of our third technique, which addresses mismatched skin tone between source avatars by shifting the colour of Gaussians to better match the skin tone of a base avatar. We concluded that our method creates authentic composite avatars as long as the avatar does not contain conflicting phenotypic and socio-cultural features.

As shown, our methods work well, laying the groundwork for future work towards an editor for composite Gaussian head avatars. Improvements can be made to the mesh binding approach in order to support larger mesh deformations. Dedicated support for the strand structure of hair would allow dynamic hair movement. Better support for novel poses of the mouth, specifically, could reduce the artefacts limiting animation of this area. Overall, this research showcases the impressive photo-realistic quality achievable by composite Gaussian head avatars, demonstrating the strong potential for this technique and opening up the door for 3D artists to create even higher fidelity 3D faces in the future.

References

[1] Volker Blanz and Thomas Vetter. "A morphable model for the synthesis of 3D faces". In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194. ISBN: 0201485605. DOI: 10.1145/311535.311556. URL: https://doi.org/10.1145/311535.311556.

[2] Jiazhong Cen et al. "Segment Any 3D Gaussians". In: *arXiv preprint arXiv:2312.00860* (2023).

[3] Yiwen Chen et al. *GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting*. 2023. arXiv: 2311.14521 [cs.CV].

[4] Bernhard Egger et al. *3D Morphable Face Models – Past, Present and Future*. 2020. arXiv: 1909.01815 [cs.CV]. URL: https://arxiv.org/abs/1909.01815.

[5] Antoine Guédon and Vincent Lepetit. "Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering". In: *ECCV* (2024).

[6] Antoine Guédon and Vincent Lepetit. "SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering". In: *CVPR* (2024).

[7] Chin-Chang Ho and Karl F. MacDorman. "Revisiting the uncanny valley theory: Developing and validating an alternative to the Godspeed indices". In: *Computers in Human Behavior* 26.6 (2010). Online Interactivity: Role of Technology in Behavior Change, pp. 1508–1518. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2010.05.015. URL: https://www.sciencedirect.com/science/article/pii/S0747563210001536.

[8] Jonathan Hoffstadt and Preston Cothren. *Dear PyGui: GPU-Accelerated GUI Toolkit for Python*. urlhttps://dearpygui.readthedocs.io. Accessed: 2025-08-18. 2025.

[9] Haoda Huang et al. "Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition". In: *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011. ISBN: 9781450309431. DOI: 10.1145/1964921.1964969. URL: https://doi.org/10.1145/1964921.1964969.

[10] Ian Jolliffe. "Principal Component Analysis". In: *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, 2014. ISBN: 9781118445112. DOI: https://doi.org/10.1002/9781118445112.stat06472. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat06472. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat06472.

[11] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70. ISBN: 3905673363.

[12] Bernhard Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics* 42.4 (July 2023). URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[13] Tobias Kirschstein et al. "NeRSemble: Multi-View Radiance Field Reconstruction of Human Heads". In: *ACM Trans. Graph.* 42.4 (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592455. URL: https://doi.org/10.1145/3592455.

[14] Samuli Laine et al. "Modular Primitives for High-Performance Differentiable Rendering". In: *ACM Transactions on Graphics* 39.6 (2020).

[15] Tianye Li et al. "Learning a model of facial shape and expression from 4D scans". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36.6 (2017), 194:1–194:17. URL: https://doi.org/10.1145/3130800.3130813.

[16] Xiaoye Sherry Li et al. "SuperLU". In: *Encyclopedia of Parallel Computing*. Springer, 2011, pp. 1955–1962.

[17] Haimin Luo et al. “GaussianHair: Hair Modeling and Rendering with Light-aware Gaussians”. In: *arXiv preprint arXiv:2402.10483* (2024).

[18] Wan-Chun Ma, Marco Barbati, and J. P. Lewis. “A facial composite editor for blendshape characters”. In: *Proceedings of the Digital Production Symposium*. DigiPro ’12. Glendale, California: Association for Computing Machinery, 2012, pp. 21–26. ISBN: 9781450316491. DOI: 10.1145/2370919.2370923. URL: https://doi.org/10.1145/2370919.2370923.

[19] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *ECCV*. 2020.

[20] Adam Paszke et al. “PyTorch: an imperative style, high-performance deep learning library”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[21] Shenhan Qian et al. “Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20299–20309.

[22] Ravi Ramamoorthi and Pat Hanrahan. “An efficient representation for irradiance environment maps”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 497–500. ISBN: 158113374X. DOI: 10.1145/383259.383317. URL: https://doi.org/10.1145/383259.383317.

[23] Shridhar Ravikumar. “Performance Driven Facial Animation with Blendshapes”. PhD thesis. University of Bath, 2018.

[24] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG]. URL: https://arxiv.org/abs/1609.04747.

[25] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[26] Johannes Lutz Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[27] Markus Schütz et al. “Splatshop: Efficiently Editing Large Gaussian Splat Models”. In: *Computer Graphics Forum* (2025). ISSN: 1467-8659. DOI: 10.1111/cgf.70214.

[28] Zhijing Shao et al. “SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2024.

[29] Georgii Stanishevskii et al. *Deepfake for the Good: Generating Avatars through Face-Swapping with Implicit Deepfake Generation*. 2024. arXiv: 2402.06390 [cs.CV].

[30] David Svitov et al. “HAHA: Highly Articulated Gaussian Human Avatars with Textured Mesh Prior”. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Dec. 2024, pp. 4051–4068.

[31] Diego Thomas and Rin-ichiro Taniguchi. “Augmented Blendshapes for Real-Time Simultaneous 3D Head Modeling and Facial Motion Capture”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[32] Pauli Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[33] Michael Weinmann and Reinhard Klein. “Advances in geometry and reflectance acquisition (course notes)”. In: *SIGGRAPH Asia 2015 Courses*. SA ’15. Kobe, Japan: Association for Computing Machinery, 2015. ISBN: 9781450339247. DOI: 10.1145/2818143.2818165. URL: https://doi.org/10.1145/2818143.2818165.

[34] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).

[35] Wojciech Zielonka et al. “Drivable 3D Gaussian Avatars”. In: *I3DV*. Mar. 2025.

A

NeRSemble Subjects

302

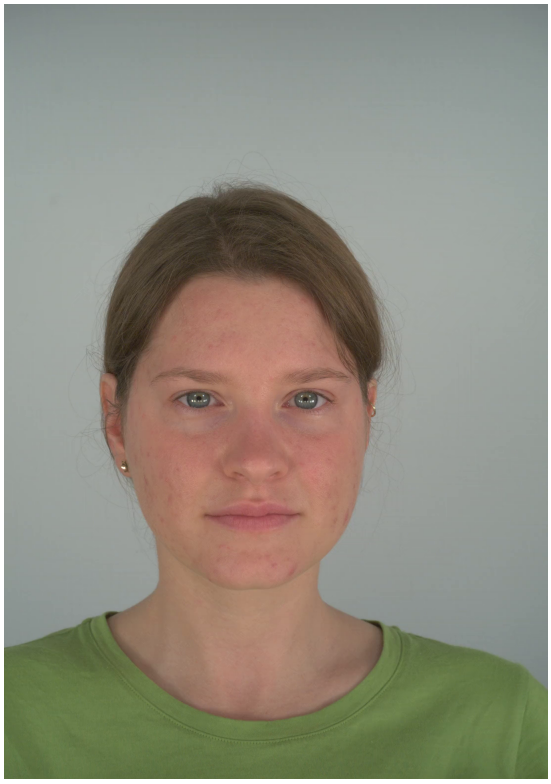
210

304

238

264

140



460

218

165

074

104

175

306

253

B

Survey Avatars

B.1 unedited_original

Avatar	1
Base	306
Authenticity	0.95
Eyes	1.00
Mouth	1.00
Nose	0.95
NHE	1.00

Avatar	2
Base	210
Authenticity	0.91
Eyes	0.95
Mouth	0.95
Nose	1.00
NHE	0.95

Avatar	3
Base	074
Authenticity	0.82
Eyes	0.95
Mouth	0.95
Nose	0.91
NHE	1.00

Avatar	4
Base	165
Authenticity	0.82
Eyes	0.91
Mouth	0.95
Nose	1.00
NHE	0.95

Avatar	5
Base	302
Authenticity	0.82
Eyes	0.91
Mouth	0.91
Nose	1.00
NHE	0.91

Avatar	6
Base	104
Authenticity	0.73
Eyes	1.00
Mouth	0.91
Nose	0.95
NHE	0.86

Avatar	7
Base	304
Authenticity	0.55
Eyes	1.00
Mouth	0.55
Nose	1.00
NHE	1.00

B.2 unedited_filling

Avatar	8
Base	253
Authenticity	0.86
Eyes	0.95
Mouth	0.86
Nose	1.00
NHE	1.00

Avatar	9
Base	460
Authenticity	0.86
Eyes	0.95
Mouth	0.95
Nose	0.95
NHE	1.00

Avatar	10
Base	218
Authenticity	0.82
Eyes	1.00
Mouth	0.95
Nose	0.86
NHE	1.00

Avatar	11
Base	140
Authenticity	0.77
Eyes	0.95
Mouth	0.95
Nose	1.00
NHE	0.82

Avatar	12
Base	175
Authenticity	0.77
Eyes	1.00
Mouth	0.77
Nose	1.00
NHE	1.00

Avatar	13
Base	238
Authenticity	0.73
Eyes	0.91
Mouth	0.91
Nose	1.00
NHE	0.82

Avatar	14
Base	264
Authenticity	0.73
Eyes	1.00
Mouth	0.73
Nose	1.00
NHE	0.95

B.3 edit_original

Avatar	15
Base	306
Eyes	104
Authenticity	0.73
Eyes	0.82
Mouth	1.00
Nose	0.95
NHE	0.91

Avatar	16
Base	074
NHE	253
Mouth	165
Authenticity	0.41
Eyes	0.82
Mouth	0.82
Nose	0.91
NHE	0.59

Avatar	17
Base	104
Eyes	460
Nose	238
Authenticity	0.32
Eyes	0.59
Mouth	0.82
Nose	0.86
NHE	0.82

Avatar	18
Base	140
NHE	238
Mouth	238
Authenticity	0.32
Eyes	0.86
Mouth	0.82
Nose	0.95
NHE	0.41

Avatar	19
Base	306
Mouth	074
Nose	074
NHE	304
Authenticity	0.27
Eyes	0.77
Mouth	0.68
Nose	0.77
NHE	0.50

Avatar	20
Base	218
Eyes	210
Nose	210
NHE	210
Authenticity	0.18
Eyes	0.59
Mouth	0.86
Nose	0.64
NHE	0.64

Avatar	21
Base	306
Mouth	074
Eyes	175
Nose	074
Authenticity	0.18
Eyes	0.18
Mouth	0.73
Nose	0.95
NHE	0.86

Avatar	22
Base	460
NHE	264
Authenticity	0.09
Eyes	0.68
Mouth	0.95
Nose	0.95
NHE	0.18

Avatar	23
Base	253
Eyes	140
Authenticity	0.09
Eyes	0.09
Mouth	1.00
Nose	0.95
NHE	0.95

Avatar	24
Base	253
NHE	302
Mouth	140
Eyes	302
Authenticity	0.09
Eyes	0.50
Mouth	0.86
Nose	0.91
NHE	0.36

Avatar	25
Base	074
Nose	210
NHE	264
Authenticity	0.09
Eyes	0.91
Mouth	0.95
Nose	0.45
NHE	0.32

Avatar	26
Base	238
Nose	253
Mouth	104
NHE	253
Authenticity	0.05
Eyes	0.73
Mouth	0.55
Nose	0.77
NHE	0.14

Avatar	27
Base	210
NHE	074
Authenticity	0.05
Eyes	0.86
Mouth	0.86
Nose	0.91
NHE	0.05

Avatar	28
Base	218
Eyes	210
Authenticity	0.00
Eyes	0.00
Mouth	1.00
Nose	0.91
NHE	0.95

Avatar	29
Base	218
Nose	460
Mouth	074
NHE	460
Authenticity	0.00
Eyes	0.73
Mouth	0.86
Nose	0.45
NHE	0.32

B.4 edit_filling

Avatar	30
Base	140
Nose	304
Authenticity	0.50
Eyes	1.00
Mouth	0.91
Nose	0.73
NHE	0.82

Avatar	31
Base	253
Nose	306
NHE	306
Authenticity	0.45
Eyes	0.86
Mouth	0.82
Nose	1.00
NHE	0.64

Avatar	32
Base	304
Nose	306
Authenticity	0.41
Eyes	0.82
Mouth	0.45
Nose	0.91
NHE	0.95

Avatar	33
Base	238
Mouth	460
Authenticity	0.32
Eyes	0.59
Mouth	0.73
Nose	0.82
NHE	0.77

Avatar	34
Base	302
Eyes	238
Authenticity	0.23
Eyes	0.32
Mouth	0.95
Nose	1.00
NHE	0.95

Avatar	35
Base	218
Mouth	140
Nose	140
Eyes	140
Authenticity	0.14
Eyes	0.77
Mouth	0.73
Nose	0.36
NHE	1.00

Avatar	36
Base	253
Eyes	302
NHE	302
Authenticity	0.09
Eyes	0.73
Mouth	0.77
Nose	0.91
NHE	0.18

Avatar	37
Base	306
NHE	238
Eyes	238
Authenticity	0.09
Eyes	0.32
Mouth	0.86
Nose	0.95
NHE	0.32

Avatar	38
Base	074
Mouth	302
Eyes	302
Authenticity	0.05
Eyes	0.09
Mouth	0.68
Nose	0.91
NHE	0.86

Avatar	39
Base	175
NHE	253
Eyes	253
Mouth	253
Authenticity	0.05
Eyes	0.50
Mouth	0.64
Nose	0.64
NHE	0.27

Avatar	40
Base	104
Eyes	175
NHE	218
Nose	218
Authenticity	0.00
Eyes	0.14
Mouth	0.64
Nose	0.77
NHE	0.27

Avatar	41
Base	460
Mouth	264
Nose	264
Authenticity	0.00
Eyes	0.91
Mouth	0.09
Nose	0.05
NHE	0.95

Avatar	42
Base	253
Mouth	175
NHE	238
Authenticity	0.00
Eyes	0.77
Mouth	0.18
Nose	0.50
NHE	0.36

Avatar	43
Base	104
Mouth	264
Nose	264
NHE	175
Authenticity	0.00
Eyes	0.64
Mouth	0.32
Nose	0.36
NHE	0.09

Avatar	44
Base	302
Mouth	264
Authenticity	0.00
Eyes	0.91
Mouth	0.05
Nose	0.86
NHE	0.77

B.5 edit_filling_smooth

Avatar	45
Base	165
NHE	306
Authenticity	0.59
Eyes	0.82
Mouth	0.91
Nose	1.00
NHE	0.82

Avatar	46
Base	140
NHE	253
Nose	253
Authenticity	0.45
Eyes	0.91
Mouth	1.00
Nose	1.00
NHE	0.50

Avatar	47
Base	460
Nose	302
NHE	302
Authenticity	0.41
Eyes	0.73
Mouth	0.86
Nose	0.95
NHE	0.64

Avatar	48
Base	210
Eyes	253
NHE	253
Authenticity	0.27
Eyes	0.73
Mouth	0.91
Nose	0.95
NHE	0.36

Avatar	49
Base	218
Mouth	306
NHE	264
Authenticity	0.14
Eyes	0.82
Mouth	0.91
Nose	0.86
NHE	0.27

Avatar	50
Base	306
NHE	264
Authenticity	0.14
Eyes	0.77
Mouth	0.77
Nose	0.73
NHE	0.32

Avatar	51
Base	074
Mouth	238
NHE	175
Authenticity	0.14
Eyes	0.82
Mouth	0.68
Nose	0.77
NHE	0.23

Avatar	52
Base	074
NHE	302
Nose	210
Authenticity	0.09
Eyes	0.95
Mouth	0.95
Nose	0.50
NHE	0.23

Avatar	53
Base	302
NHE	104
Eyes	218
Nose	104
Authenticity	0.09
Eyes	0.64
Mouth	0.86
Nose	0.77
NHE	0.18

Avatar	54
Base	264
Eyes	104
NHE	104
Mouth	175
Authenticity	0.05
Eyes	0.23
Mouth	0.77
Nose	0.86
NHE	0.27

Avatar	55
Base	460
Eyes	264
Nose	104
Mouth	264
Authenticity	0.05
Eyes	0.18
Mouth	0.09
Nose	0.86
NHE	0.91

Avatar	56
Base	175
Mouth	264
Eyes	238
NHE	264
Authenticity	0.00
Eyes	0.27
Mouth	0.91
Nose	0.68
NHE	0.73

Avatar	57
Base	238
Mouth	218
Authenticity	0.00
Eyes	0.50
Mouth	0.09
Nose	0.68
NHE	0.82

Avatar	58
Base	460
Nose	165
Mouth	165
Authenticity	0.00
Eyes	0.91
Mouth	0.09
Nose	0.55
NHE	1.00

Avatar	59
Base	175
Mouth	140
NHE	460
Authenticity	0.00
Eyes	0.68
Mouth	0.41
Nose	0.64
NHE	0.36

C

Excluded Avatars

Avatar 16 **Reason** Skin tone		**Avatar** 18 **Reason** Gender	
Avatar 19 **Reasons** Gender Skin tone		**Avatar** 21 **Reason** Skin tone	
Avatar 22 **Reason** Skin tone		**Avatar** 24 **Reason** Gender	
Avatar 25 **Reasons** Gender Skin tone		**Avatar** 26 **Reason** Gender	

Avatar 27
Reason
Skin tone

Avatar 29
Reasons
Gender
Skin tone

Avatar 36
Reason
Gender

Avatar 37
Reason
Gender

Avatar 38
Reason
Skin tone

Avatar 39
Reasons
Subject 175
Gender
Skin tone

Avatar 40
Reasons
Subject 175
Skin tone

Avatar 41
Reason
Skin tone

Avatar 42
Reasons
Subject 175
Gender
Skin tone

Avatar 43
Reasons
Subject 175
Gender
Skin tone

Avatar 44
Reason
Skin tone

Avatar 49
Reasons
Gender
Skin tone

Avatar 50
Reasons
Gender
Skin tone

Avatar 51
Reasons
Subject 175
Gender
Skin tone

Avatar 52
Reasons
Gender
Skin tone

Avatar 53
Reason
Gender

Avatar 54
Reasons
Subject 175
Gender
Skin tone

Avatar 55
Reason
Skin tone

Avatar 56
Reasons
Subject 175
Skin tone

Avatar 57
Reason
Gender

Avatar 58
Reason
Gender

Avatar 59
Reasons
Subject 175
Gender
Skin tone