Efficient 3D Implicit Head Avatar with Mesh-anchored Hash Table Blendshapes

Ziqian Bai[1,2*] Feitong Tan[1] Sean Fanello[1] Rohit Pandey[1]
Mingsong Dou[1] Shichen Liu[1] Ping Tan[3] Yinda Zhang[1]
[1] Google [2] Simon Fraser University [3] The Hong Kong University of Science and Technology

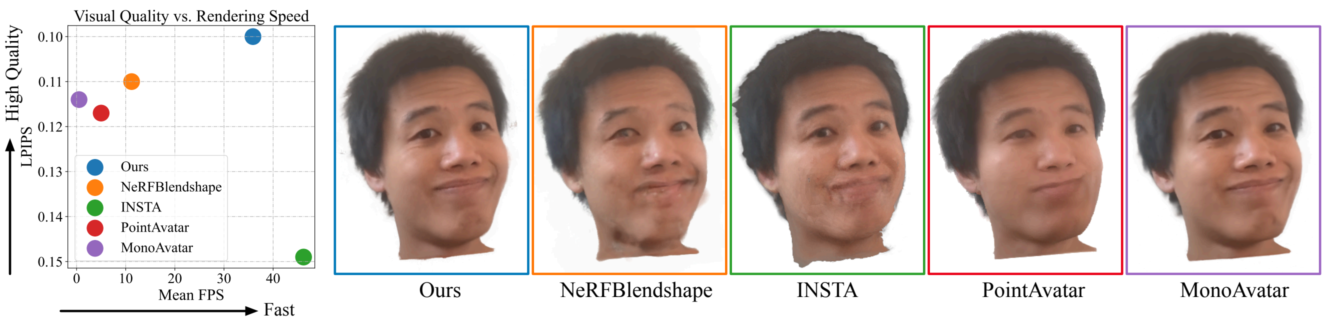

Figure 1. Visual Quality vs. Rendering Speed comparisons between our monocular 3D head avatars and prior state-of-the-art methods including NeRFBlendshape [14], INSTA [42], PointAvatar [41] and MonoAvatar [5]. Our approach achieves real-time rendering (*i.e.* > 30 mean FPS) with a 512×512 resolution, while produces comparable visual quality comparing to prior SOTAs [5], and gives significantly better results on challenging expressions than prior efficient avatars [14, 41, 42]. Webpage augmentedperception.github.io/monoavatar-plus

Abstract

3D head avatars built with neural implicit volumetric representations have achieved unprecedented levels of photorealism. However, the computational cost of these methods remains a significant barrier to their widespread adoption, particularly in real-time applications such as virtual reality and teleconferencing. While attempts have been made to develop fast neural rendering approaches for static scenes, these methods cannot be simply employed to support realistic facial expressions, such as in the case of a dynamic facial performance. To address these challenges, we propose a novel fast 3D neural implicit head avatar model that achieves real-time rendering while maintaining fine-grained controllability and high rendering quality. Our key idea lies in the introduction of local hash table blendshapes, which are learned and attached to the vertices of an underlying face parametric model. These per-vertex hash-tables are linearly merged with weights predicted via a CNN, resulting in expression dependent embeddings. Our novel representation enables efficient density and color predictions using a lightweight MLP, which is further accelerated by a hierarchical nearest neighbor search method. Extensive experiments show that our approach runs in real-time while achieving comparable rendering quality to state-of-the-arts and decent results on challenging expressions.

1. Introduction

The demand of high performing photo-realistic human avatars has dramatically increased with emerging VR/AR applications, *e.g.* VR gaming [2, 33], virtual assistant [3], tele-presence [28], and 3D videos [15, 26]. How to build efficient high quality avatars from monocular RGB videos becomes a promising direction due to the convenience of monocular data acquisition. While early works mostly adopt surface-based models for convenient controllability, recent methods (*e.g.* MonoAvatar [5]) leverage a sophisticated pipeline to build human avatars on neural radiance fields, which delivers vivid animations as well as significantly better rendering quality, especially over challenging parts such as hairs and glasses. On the downside, these approaches tend to be prohibitively slow, and most of the computation is consumed by the neural radiance field inference with large Multilayer Perceptrons (MLPs).

Recently, fast approaches for neural radiance fields (*e.g.* hash encoding in Instant NGPs [27]) have been proposed, which are designed mostly for static scenes or pre-recorded temporal sequences. Despite their great success, it is not straightforward to extend these approaches for human avatars, which requires real-time rendering of dynamic facial performances when controlling the avatar. NeRFBlend-

*Work was conducted while Ziqian Bai was an intern at Google.

shape [14] address these issues by learning multiple feature hash tables, one for each face blendshape. These hash tables are linearly combined with blending weights to render the target facial expression via hash encoding. However, the expressiveness of the avatar is compromised by the global nature of the blending shapes, which cannot accurately capture vertex-level local deformations. On the other hand, INSTA [42] proposes to build the appearance model in a canonical space using a vanilla Instant NGPs [27] with expression codes as the additional MLP input to capture dynamic details, which is then transformed into target expression via a face parametric model (*i.e*. 3D morphable model (3DMM)). However, the lightweight MLP used in the vanilla Instant NGPs limits their model capacity, resulting in inferior animation quality, especially on extreme expressions.

In this work, we propose a novel 3D neural avatar system that achieves efficient inference while maintaining fine-grained controllability and high fidelity quality. To achieve this, we introduce mesh-anchored hash table blendshapes, where we attach multiple, small hash tables to each of the 3DMM mesh vertices. These hash tables act as per-vertex "local blendshapes" (*i.e*., each "blendshape" is controlled by one local hash table) and influence only a local region. The mesh-anchored blendshapes are linearly merged with per-vertex weights predicted by a convolutional neural network in UV space from avatar driving signals, such as expression and head rotation. This results in expression-dependent hash table embeddings which offer several advantages over a global linear combination of blendshapes. Indeed by associating hash tables with individual vertices, we enhance the expressiveness of the model, allowing for more localized and nuanced facial expressions. This contrasts with global blendshapes, which apply uniform transformations across the entire face, limiting expressiveness.

In more detail, our model starts from 3D query points, uses hash encoding [27] to gather the merged hash table embeddings from k-nearest-neighbor vertices around the query point, and predicts the density and color via a small MLP. The hash encoding [27] allows us to use a very lightweight MLP to significantly reduce computation, leading to efficient inference. Additionally, the vertex-attached hash table blendshapes represent a 3DMM-anchored neural radiance field (NeRF), which can be easily controlled by the underlying 3DMM and produce high fidelity renderings as demonstrated by MonoAvatar [5]. To further accelerate our rendering speed, we propose a hierarchical k-nearest-neighbor search method.

Our contributions are summarized as follows. We propose a novel approach for high quality and efficient 3D neural implicit head avatars. At the core of our model, vertex-attached local hash table blendshapes are proposed to support efficient rendering, controllability, and capturing fine-grained rendering details in dynamic facial performances. We also design a hierarchical querying solution to speed up the k-nearest-neighbor search when pulling hash table embeddings from neighbor vertices. Extensive experiments on multiple datasets verify that we are able to speed up avatar rendering to real-time (*i.e*., average $\geq$ 30 FPS to render a 512×512 video) while maintaining comparable rendering quality with the state-of-the-art high quality 3D avatar [5] and being largely superior on challenging expressions than existing efficient 3D avatars [14, 41, 42].

2. Related Work

Constructing photorealistic digital humans has been a extensively researched topic. Here, we focus on discussing prior work on implicit monocular head avatars and efficient rendering. We refer readers to state-of-the-art surveys [12, 32, 43] for a comprehensive literature review.

High Quality Head Avatar. Traditionally, high-quality head avatars have been achieved under expensive equipment configurations, such as camera arrays [6, 23, 28], depth sensors [7], and light stages [15, 25], or require laborious manual intervention [1]. Recent research efforts have focused on constructing high quality avatars from monocular RGB videos. One typical class of approaches [4, 13, 35, 40] use implicit 3D representations (*i.e*., neural radiance fields (NeRFs), implicit occupancy fields) to build the head avatar, which are parameterized by Multilayer perceptrons (MLPs). Although reasonable results are obtained, their rendering quality is still unsatisfactory especially for more challenging expressions. More recently, Bai *et al*. [5] proposed a head avatar based on 3DMM-anchored NeRFs with expression-dependent features produced by a convolution neural network in UV space. Chen *et al*. [9] designed local deformation fields to capture expression-dependent deformations applied on canonical NeRFs. Despite the impressive results, their methods didn't demonstrate real-time rendering capability due to expensive inference using large MLPs. In contrast, with our proposed mesh-anchored hash table blendshapes (Sec. 3), we achieve much faster rendering speed while maintaining high fidelity results.

Efficient Neural Radiance Fields. There has been a plethora of work in recent years attempting to accelerate rendering with neural implicit representations for static objects and scenes. SNeRG [22], DVGO [31] and Plenoxels [37] propose to directly optimize voxel grids of (neural or SH) features for faster performance. However, their approach still requires a large memory footprint to store per-voxel features in 3D space. KiloNeRF [30] dramatically accelerates the original NeRF by representing the scene with thousands of tiny MLPs, however, this approach requires a complex training strategy. TensoRF [8] factorizes the feature grid into compact components, resulting in sig-

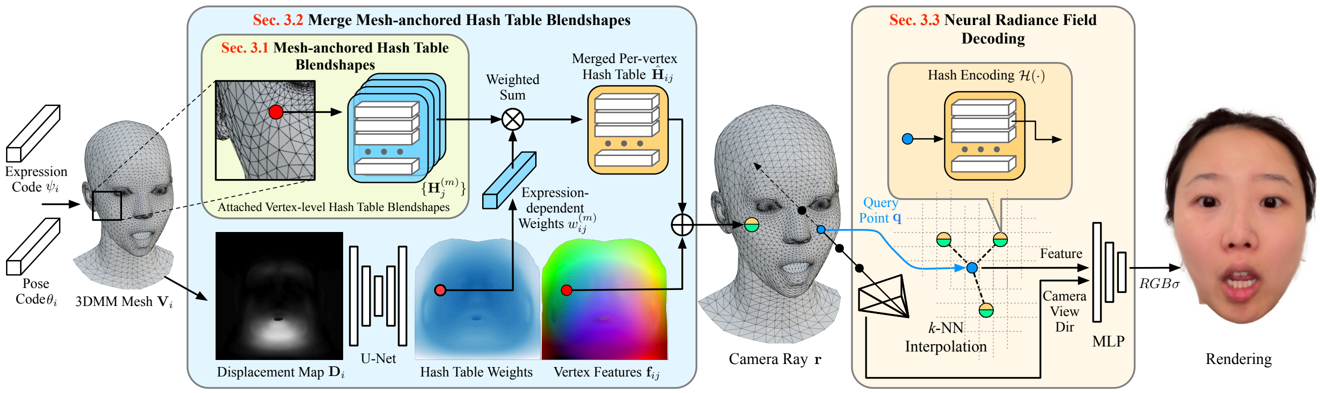

Figure 2. Overview of our pipeline. Our core avatar representation is Mesh-anchored Hash Table Blendshapes (Sec. 3.1), where multiple small hash tables are attached to each 3DMM vertex. During inference, our method starts from a displacement map encoding the facial expression, which is then fed into a U-Net to predict hash table weights and per-vertex features. The predicted weights are used to linearly combine the multiple hash tables attached on each 3DMM vertex (Sec. 3.2). During volumetric rendering (Sec. 3.3 and Fig. 3), for each query point, we search its k-nearest-neighbor vertices, then pull embeddings from the merged hash tables and concatenate with the per-vertex feature to decode local density and color via a tiny MLP with two hidden layers.

nificantly higher memory efficiency. Concurrently, Instant NGPs [27] utilizes multi-resolution hashing for efficient encoding, resulting in high compactness. MobileNeRF [10] propose to represent NeRF based on polygons, which allows leveraging a traditional polygon pipeline to enable their method to run in real-time on mobile devices. 3D Gaussian Splatting [17] represents a radiance field with 3D Gaussian point clouds, and leverages a point rasterization pipeline to enable fast rendering. These methods, however, cannot be easily extended to controllable dynamic contents. More recently, NeRFBlendshape [14] was proposed to handle controllable expressions by learning multiple hash tables for different global blendshapes and linearly combine them with expression codes. INSTA [42] transforms all expressions into a shared 3D canonical space, then adopts the vanilla Instant NGPs [27] conditioned on expression codes to model the head avatar. Despite the fast rendering speed achieved, their methods suffer from unsatisfactory rendering quality. PointAvatar [41] utilizes point clouds to represent the head avatar and uses large MLPs to predict the colors and motions of each point, leading to slow inference. Another type of works [11, 18, 19, 34, 36, 39] use 2D convolution neural networks to directly synthesize images (*i.e.* 2D neural rendering) from rasterized 3DMM meshes or low resolution feature maps generated by volumetric rendering. Despite their fast speed, the 2D CNNs may break the 3D consistency, leading to temporally unstable results especially for high frequency details. In contrast, our method simultaneously achieves controllability, high quality, and efficient rendering with a fully 3D representation.

3. Method

Given a monocular RGB video, our method learns a neural radiance field (NeRF) based head avatar, which can be rendered under any specified cameras, articulated poses (*i.e.*, neck, jaw, and eyes) and facial expressions defined by a face parametric model (*i.e.* 3DMM). We use FLAME [20] as the parametric model in this work, but our method can be generalized to any other mesh-based parametric models. Fig. 2 shows the overview of our method.

Our goal is to design a 3D neural implicit head avatar architecture that can simultaneously achieve high image quality, controllability, and computationally efficient rendering. To achieve this, we propose mesh-anchored hash table blendshapes (Sec. 3.1) as a novel avatar representation that can leverage both advantages from recent high-quality (*i.e.*, 3DMM-anchored NeRF [5]) and efficient (*i.e.*, hash encoding [27]) frameworks. More specifically, we propose to attach multiple small hash tables on each 3DMM vertex. These vertex-attached hash tables form a set of local "blendshapes", which will be linearly merged with predicted blending weights (Sec. 3.2), and decoded into a 3DMM-anchored NeRF to support fine-grained control and high fidelity rendering. During NeRF decoding on a query point (Sec. 3.3), we pull the embeddings from the linearly merged hash tables attached on k-nearest-neighbor (k-NN) vertices from the 3DMM mesh. Using hash encoding allows us to use a very light weight MLP (only 2 hidden layers) to predict the final densities and colors, which is the key for efficient rendering. To further accelerate our approach to real-time, we leverage the fact that close query points likely share similar k-NN vertices, and thus propose to group the query points into voxels and hierarchically search for k-NN vertices (Sec. 3.4). Finally, our proposed avatar representation can be trained with only monocular RGB videos without any 3D scans or multi-view data (Sec. 3.5).

3.1. Mesh-anchored Hash Table Blendshapes

The core of our model is an avatar representation that can represent a 3DMM-anchored neural radiance field (NeRF) while allowing us to adopt the hash encoding [27] technique for acceleration. Recent approaches manage to adopt hash encoding into head avatars with different avatar representations (*i.e.*, global blendshapes [14] and canonical NeRF [42]). In contrast, our solution is built upon the most recent 3DMM-anchored NeRF [5], which is superior for high quality renderings as demonstrated in our experiments (Sec. 4.2).

We propose mesh-anchored hash table blendshapes as the new avatar representation. Given a target expression i with 3DMM pose code $\boldsymbol{\theta}_i$ and expression code $\boldsymbol{\psi}_i$, we first get the deformed 3DMM mesh $\mathbf{V}_i = \mathcal{F}_{\text{3DMM}}(\boldsymbol{\psi}_i, \boldsymbol{\theta}_i)$ with J vertices. For the j-th 3DMM mesh vertex $\mathbf{v}_{ij}$, we attach M small hash tables $\{\mathbf{H}_j^{(m)}\}_M$ on it, where each hash table has multiple resolutions following instant NGPs [27]. Intuitively, these hash tables form a set of vertex-level "blendshapes" anchored on the mesh, where each "blendshape" is a hash table, whose embeddings encode the information of a local radiance field around vertex $\mathbf{v}_{ij}$. Given a target expression to render, these hash table blendshapes are linearly summed via expression-dependent weights (Sec. 3.2), such that the merged embeddings encode the fine details specific to the target expression. Simultaneously, the coarse motion of the target expression is captured by the 3DMM vertex movement, which moves the attached hash tables accordingly, and hence the corresponding local radiance field.

3.2. Merge Mesh-anchored Blendshapes

We obtain per-vertex blending weights by running a convolution neural network (CNN) on the 3DMM deformation represented in UV atlas space. Specifically, we calculate the vertex displacements with respect to the neutral face $\mathbf{D}_i = \mathcal{F}_{\text{3DMM}}(\boldsymbol{\psi}_i, \boldsymbol{\theta}_i) - \mathcal{F}_{\text{3DMM}}(\mathbf{0}, \mathbf{0})$. The displacements are then warped into the UV space and fed into a U-Net to predict a weights map in $\mathbb{R}^{\mathrm{H_t} \times \mathrm{W_t} \times \mathrm{M}}$, where $\mathrm{H_t} \times \mathrm{W_t}$ is the UV resolution, and M is the number of hash table blendshapes on each vertex (pre-defined as 5 in our experiments). The weights map is then sampled back to 3DMM vertices, serving as the expression-dependent weights $\{w_{ij}^{(m)}\}_M$ to take a weighted sum of the embeddings in the hash tables on each vertex, which produces the merged hash tables

$$\hat{\mathbf{H}}_{ij} = \sum_{m=1}^{M} w_{ij}^{(m)} \mathbf{H}_j^{(m)}. \tag{1}$$

The U-Net also produces a UV feature map. We sample a per-vertex feature $\mathbf{f}_{ij}$ from it, similar to MonoAvatar [5]. We empirically found this benefits the geometry quality. The mesh-anchored hash tables $\hat{\mathbf{H}}_{ij}$ and features $\mathbf{f}_{ij}$ are decoded into a neural radiance field as described in Sec. 3.3.

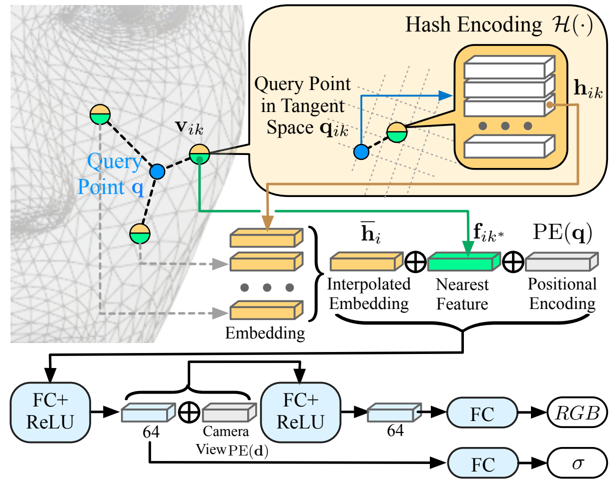

Figure 3. Pipeline of Neural Radiance Field Decoding (Sec. 3.3). Our method only needs a very lightweight MLP for NeRF inference, where a two-hidden-layer MLP is used to predict the density and the color of each query point.

3.3. Neural Radiance Field Decoding

Given the mesh-anchored hash tables $\hat{\mathbf{H}}_{ij}$ and features $\mathbf{f}_{ij}$ described in Sec. 3.2, the final step is to decode them into a Neural Radiance Field (NeRF) to render the output image as shown in Fig. 3. The key idea is to associate a query point to neighbor vertices, and pull the embeddings from the attached hash tables via hash encoding [27]. Finally, we decode these pulled embeddings and the nearest per-vertex feature into the color and the density of this query point, followed by volumetric rendering to obtain the output image.

For a 3D query point $\mathbf{q}$ when rendering a particular facial expression i, we first obtain its k-nearest-neighbors, denoted as $\{\mathbf{v}_{ik}\}_{k \in \mathcal{N}_{\mathbf{q}}^K}$, from the 3DMM vertices, with k^* denoting the nearest vertex index. For each neighbor vertex $\mathbf{v}_{ik}$ with an attached hash table $\hat{\mathbf{H}}_{ik}$, we denote $\mathbf{q}_{ik}$ as the coordinates of $\mathbf{q}$ in the tangent space of $\mathbf{v}_{ik}$. We then use $\mathbf{q}_{ik}$ to query the hash table $\hat{\mathbf{H}}_{ik}$ using a hash encoding function $\mathcal{H}(\cdot)$ and obtain the embedding $\mathbf{h}_{ik} = \mathcal{H}(\mathbf{q}_{ik}; \hat{\mathbf{H}}_{ik})$. To interpolate the embeddings from all k-nearest-neighbors, we use the weighted sum of the inverse distances $z_k = 1/\|\mathbf{q}_{ik}\|_2$. Next, the summed embedding, together with the nearest per-vertex feature $\mathbf{f}_{ik^*}$ and the query point tangent coordinate $\mathbf{q}_{ik^*}$ of the nearest vertex, are fed into a two-hidden-layer MLP to predict the density and color as

$$\bar{\mathbf{h}}_i = \sum_{k \in \mathcal{N}_{\mathbf{q}}^K} \overline{w}_k \mathbf{h}_{ik}, \text{where } \overline{w}_k = \frac{z_k}{\sum_{k' \in \mathcal{N}_{\mathbf{q}}^K} z_{k'}}, \tag{2}$$

$$\Big[\mathbf{c}_i(\mathbf{q}, \mathbf{d}), \sigma_i(\mathbf{q})\Big] = \mathcal{F}_{\text{MLP}}\left(\bar{\mathbf{h}}_i, \mathbf{f}_{ik^*}, \text{PE}(\mathbf{q}_{ik^*}), \text{PE}(\mathbf{d})\right),$$

where $\overline{w}_k$ is the normalized inverse-distance based weight, $\mathbf{d}$ denotes the camera view direction, $\text{PE}(\cdot)$ denotes positional encoding, $\mathbf{c}_i$ denotes color, and σ_i denotes density.

Finally, we render the output pixel with the given camera ray $\mathbf{r}$ by volumetric rendering, where we reparameterize the query point with samples on the ray $\mathbf{q} = \mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{C}_i(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma_i(\mathbf{r}(t))\mathbf{c}_i(\mathbf{r}(t), \mathbf{d})dt,$$
$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma_i(\mathbf{r}(s))ds\right). \tag{3}$$

Following prior works [5, 16], we also introduce a per-frame error-correction warping field during training to reduce misalignments due to the noise in 3DMM tracking and unmodeled per-frame contents such as hair movements. We feed the query point $\mathbf{q}$, together with a per-frame latent code $\mathbf{e}_i$, into an MLP $\mathcal{F}_{\mathcal{E}}(\cdot)$ to obtain a rigid transformation applied on the original query point, denoted as $\mathbf{q}' = \mathcal{T}_i(\mathbf{q}) = \mathcal{F}_{\mathcal{E}}(\mathbf{q}, \mathbf{e}_i)$. The warped query point $\mathbf{q}'$ is then used to compute the density and color for volumetric rendering. Since the warping fields are overfit to corresponding training frames, we disable the warping field during testing similar to previous works [5, 16], and hence $\mathcal{F}_{\mathcal{E}}$ does not affect rendering efficiency.

3.4. Hierarchical k-nearest-neighbor Search

As described in Sec. 3.3, our method involves a k-nearest-neighbor (k-NN) search, which is computationally expensive and cannot be naively accelerated with pre-calculated structures (*e.g.*, KD-Tree) due to the dynamically changing search pool (*i.e.*, the 3DMM vertices driven by poses and expressions). To speed up the process, we propose a hierarchical k-NN search algorithm following a coarse-to-fine strategy. The key idea is to group nearby query points into a cluster as they likely share similar nearest neighbors. Specifically, we use a 3D grid with resolution 64 and treat all query points that fall in each voxel as a cluster. For each cluster, we first search K' (where $K' > K$) nearest neighbors of the voxel center from all 3DMM vertices. Then, for each query point, we search K nearest neighbors from the K' nearest neighbors of the corresponding cluster. In practice, for a 3DMM with the vertices number of $J = 1772$, we set $K' = 12$ and $K = 3$. Our experiments empirically show that, with a proper grid resolution, this design significantly improves the nearest neighbor search speed, and does not introduce noticeable rendering artifacts, even though the k-NNs may not be accurate on some of the query points.

3.5. Training

Only monocular RGB videos are required to train our model. Three losses are used during the training process: (1) a photometric loss that minimizes the l_2-norm distance between the rendered and ground truth pixel colors over all camera rays $\mathbf{r}$ from all training frames i. Formally, we have $\mathcal{L}_{\text{rgb}} = \sum_i \sum_{\mathbf{r}} \|\mathbf{C}_i(\mathbf{r}) - \mathbf{I}_i(\mathbf{r})\|_2$; (2) a elastic regularization loss $\mathcal{L}_{\text{elastic}}$ applied on the learned error-correction warping field $\mathcal{T}(\mathbf{q})$, which is introduced in Nerfies [29]; (3) a magnitude regularization loss to encourage small warping fields, which is defined as $\mathcal{L}_{\text{mag}} = \sum_{\mathbf{q}} \|\mathbf{q} - \mathcal{T}(\mathbf{q})\|_2^2$. Finally, we combine all three loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{elastic}}\mathcal{L}_{\text{elastic}} + \lambda_{\text{mag}}\mathcal{L}_{\text{mag}}, \tag{4}$$

where we set $\lambda_{\text{elastic}} = 10^{-4}$ at the beginning of the training and decay it to 10^{-5} after 150k iterations, and set $\lambda_{\text{mag}} = 10^{-2}$. To warm start training, we replace the l_2-norm distance in the photometric loss $\mathcal{L}_{\text{rgb}}$ with the l_2 distance for the first 10k iterations. Please refer to the supplementary for more details.

4. Experiments

In this section, we first introduce the data and metrics used for training and evaluation (Sec. 4.1). Then, we show that our avatar model achieves real-time rendering speed, while producing superior rendering quality on challenging expressions than recent efficient avatars [14, 41, 42] and being comparable to previous high-quality approaches [5] (Sec. 4.2). Finally, we provide ablation studies to justify the design choices and hyper-parameters of our avatar representation, and demonstrate the rendering speed improvements contributed from each of our newly proposed algorithmic components.

4.1. Data and Metrics

Data. We use monocular RGB videos of multiple subjects (*i.e.*, one video for one subject) to train and evaluate our method, and compare to prior state-of-the-art (SOTA) approaches. Our dataset consists of 10 videos in total, which are a mix of videos captured by us, as well as videos from prior works including PointAvatar [41], INSTA [42], and MonoAvatar [5]. We filter out the background of the videos with off-the-shelf segmentation [24] and matting [21] methods, then crop and resize the videos into a VGA resolution that preserves the original aspect ratio. We compute the camera and 3DMM parameters from the videos following the 3DMM fitting optimization used in INSTA [42]. We reserve a short clip from the end of each video as the testing frames, and use the rest frames for training.

Metrics. Following prior arts [13], we use PSNR, SSIM (higher is preferable), and LPIPS (lower is preferable) to measure the image quality. As observed by Zhang *et al.* [38], LPIPS is a more effective metric in judging the perceptual quality compared to PSNR and SSIM. When computing PSNR and SSIM, we weigh the mean squared error map and the SSIM map with a foreground mask (eroded and smoothed), in order to focus on non-empty areas and avoid the inaccurate foreground segmentation from dominating the metrics.

To evaluate the computational cost, we measure the rendering speed in frames-per-second (FPS) on a RTX3090Ti and compare across different approaches with their available implementations. We also estimate the number of FLOPs (floating-point operations) of all methods as the theoretical measurement for the rendering speed. When estimating FLOPs, we fix the contribution of the ray-marching part to 16 points sampled along each camera ray. This simplifies the estimate, since the ray-marching varies the number of FLOPs needed across cameras and scenes, and it applies to all the considered methods.

4.2. Comparison to State-of-the-art

We compare our method with several prior works, including: NeRFBlendshape [13], INSTA [42], PointAvatar [41], and MonoAvatar [5]. NeRFBlendshape [13] and INSTA [42] adopt hash encoding [27] into head avatars, leading to efficient renderings. PointAvatar [41] leverages point clouds to represent the head avatar. MonoAvatar [5] is based on a 3DMM-anchored NeRF, and produces high-quality renderings but is slow in speed. We use the same camera and 3DMM parameters to train and test all methods.

From Tab. 1, ours and INSTA [42] are the only 2 methods that can achieve real-time rendering (*i.e.*, $\geq$ 30 mean FPS). However, INSTA is quantitatively inferior than our method by a large margin, and gives obvious artifacts in Fig. 4, especially for challenging expressions. PointAvatar [41] has the potential to run in real-time with an optimized implementation thanks to its point cloud representation, but their renderings are overall blurrier than ours, leading to worse quantitative results. Although NeRFBlendshape [14] gives relatively good numbers in Tab. 1, it produces severe artifacts in dynamically changing regions (*e.g.*, mouth and eyebrows in Fig. 4) for several median and large expressions and also gives more floaters, resulting in implausible animations. We highly suggest readers to see the supplementary videos for more comparisons. MonoAvatar [5] gives good rendering qualities and animations, but is one order of magnitudes slower and slightly blurrier on high frequency details such as forehead wrinkles, presumably because that the hash encoding in our model can better capture high frequency contents. Among these compared approaches, our method is the only one that achieves real-time rendering while being one of the best on image qualities.

We also compare the theoretical FLOPs of all methods in Tab. 1, where our method requires the least computation mostly because of the smaller MLPs we use (*e.g.*, 2 hidden layers of ours vs. $\geq$ 5 hidden layers of others). Note that INSTA [42] is implemented with a highly optimized pure C++ and CUDA codebase, while other methods use python (tensorflow/pytorch) with customized CUDA kernels. This implementation advantage of INSTA makes it running in a high FPS even with a relatively larger FLOPs.

	LPIPS	SSIM	PSNR	Mean FPS	GFLOPs
PointAvatar [41]	0.117	0.728	21.12	5.0	933
INSTA [42]	0.149	0.758	22.12	**46.2**	266
NeRFBlendshape [14]	0.110	0.793	**22.77**	11.2	223
MonoAvatar [5]	0.114	**0.798**	22.74	0.5	2385
Ours	**0.100**	0.795	**22.77**	**35.9**	**113**

Table 1. Quantitative comparisons with state-of-the-art approaches. Our method achieves rendering quality among the best, while supports real-time rendering with a 512×512 resolution.

4.3. Ablation Study

In this section we show the impact of the proposed design choices, in particular proving the importance of mesh-anchored hash table blendshapes and the proposed hierarchical k-NN search.

4.3.1 Mesh-anchored Hash Table Blendshapes

We hereby investigate alternative design choices and different hyper-parameters to justify the necessity of our mesh-anchored hash table blendshapes.

Static Hash + 3DMM Param. We first build a naive alternative approach to incorporate hash encoding into 3DMM-anchored NeRF, by attaching a single hash table to each vertex. For a query point, we concatenate the 3DMM pose and expression codes $(\boldsymbol{\theta}_i, \boldsymbol{\psi}_i)$ with the embedding pulled from the hash tables, and send them into the MLP. Note that there is no convolution running in UV space. As shown in Tab. 2 and Fig. 5, obvious rendering artifacts show up, and the rendering quality metrics drop significantly compared to our full model. This is presumably because the lightweight MLP, which is crucial for good efficiency, does not have enough capacity to process detailed expression-dependent information from compact 3DMM codes.

Static Hash + UV CNN. We then increase the model capacity by adding back the UV CNN branch, but use only one hash table per-vertex, *i.e.*, single hash table without blendshape formulation. As shown in Tab. 2 and Fig. 5, the rendered images show less artifacts over the previous case, but still contain blurry textures and floaters compared to our full model. This demonstrates that the blendshape formulation of mesh-anchored hash tables are necessary in order to obtain good expression dependent local embeddings, leading to a superior rendering quality.

Number of Hash Table Blendshapes. Here we investigate how the number of hash table blendshapes per vertex influences the final rendering quality. As shown in Tab. 2, we can see that more blendshapes per vertex leads to higher rendering qualities, which saturates as the number of tables increases. In Fig. 5, increasing the number of blendshapes also gives better details especially for eyelids and ears. Although further adding more blendshapes may produce a better quality, we choose to use 5 blendshapes per-vertex as our

Figure 4. Comparisons on the rendering quality to previous state-of-the-art methods. From left to right, each column contains the images of: 1) PointAvatar [41], 2) INSTA [42], 3) NeRFBlendshape [14], 4) MonoAvatar [5], 5) Ours, 6) Ground Truth. Our method faithfully reconstructs the personalized expressions and high-frequency details, achieving one of the best rendering quality with real-time rendering speed.

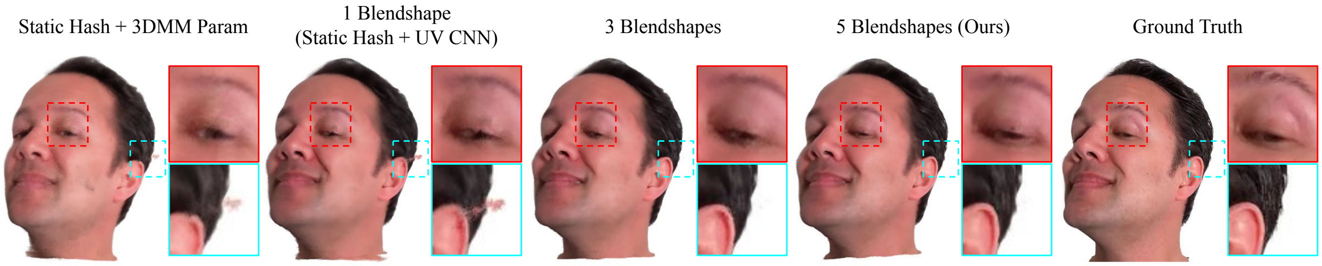

Figure 5. Qualitative results of our full model and various alternative design choices to demonstrate the necessity of our mesh-anchored hash table blendshapes. Please refer to Sec. 4.3.1 for details of each alternative design choice. "*Static Hash + 3DMM Param*" gives unfaithful expressions and obvious artifacts around eyes and ears. "*Static Hash + UV CNN*" produces more faithful expressions, but still suffers from floaters. By utilizing the blendshape formulation of hash tables, "*3 Blendshapes*" eliminates most of the artifacts, but produces blurriness on the details of eyes and ears. "*Ours*" gives the best rendering quality by increasing the number of blendshapes to 5, leading to cleaner details of the eyelids and ear boundary.

Figure 6. Qualitative results of using different 3D grid resolutions during hierarchical k-NN search and without hierarchical k-NN search for ablation studies. "*16 grid resolution*" results in blocky artifacts around mouth and nose. "*32 grid resolution*" removes blocky artifacts, but produces floaters inside mouth. "*64 grid resolution*" produces a similar level of visual quality as "*w/o hierarchical k-NN search*".

final setting to maintain a relatively small model size and computation cost.

4.3.2 Hierarchical k-NN Search

We evaluate our model with and without hierarchical k-NN search in terms of speed and quality. From the comparison over rendering speeds (w/: 35.9 FPS; w/o: 26.4 FPS), we can see that hierarchical k-NN search gives around 36% improvements on the frame rate, which is crucial for achieving real-time rendering. From Fig. 6, we empirically find that enabling hierarchical k-NN search will not lead to observable drops on the rendering quality, as long as a proper grid resolution is used (*i.e*., 64 in our case).

We also investigate the affect of using different 3D grid resolutions during hierarchical k-NN search. As shown in Fig. 6, we observe more artifacts around the mouth region when using smaller 3D grid resolutions (*i.e*., 32 and 16). Therefore, we choose to use 64 resolution in our final setting, which is a good trade-off between quality and speed.

	LPIPS	SSIM	PSNR
Static Hash + 3DMM Param	0.125	0.763	21.99
1 Blendshape (Static Hash + UV CNN)	0.115	0.785	22.52
3 Blendshapes	0.104	0.791	22.70
5 Blendshapes (Ours)	**0.100**	**0.795**	**22.77**

Table 2. Quantitative results for the ablation study. Our full model (*i.e*., mesh-anchored hash table blendshapes) consistently outperforms alternative design choices discussed in Sec. 4.3.1.

5. Discussion

We present a high quality 3D neural volumetric head avatar that can be rendered efficiently, while only requires monocular RGB videos for construction. We propose the mesh-anchored hash table blendshapes as our avatar representation, which enable a significantly faster rendering speed by utilizing hash encoding and lightweight MLPs, while still maintaining superior controllability to support realistic facial animations, and producing vivid expression-dependent details thank to the local blendshape formulation of hash tables. The experiments indicate that our approach runs in real-time at a 512×512 resolution, while giving a rendering quality comparable to state-of-the-art, with better challenging expressions than prior efficient approaches.

As a limitation, we observe floaters under camera viewpoints and expressions that are far from the training distribution, which is a common issue in instant NGPs [27] based approaches. We also notice that performance around the mouth interior regions tends to be less stable because of the relatively poor tracking in these areas on the training data. Fortunately, the fast rendering could enable the possibility to adopt more expensive training strategies, such as regularization terms, adversarial loss, or joint face fitting refinement during the training, which could potentially mitigate these issues and further improve the rendering expressiveness and quality.

References

[1] MetaHuman - Unreal Engine. https://www.unrealengine.com/en-US/metahuman. Accessed: 2022-10-17. 2

[2] VRChat. https://hello.vrchat.com. Accessed: 2023-02-05. 1

[3] XHolo Virtual Assitant. https://www.digalix.com/en/virtual-assistant-augmented-reality-xholo. Accessed: 2023-03-01. 1

[4] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. RigNeRF: Fully Controllable Neural 3D Portraits. pages 20364–20373, 2022. 2

[5] Ziqian Bai, Feitong Tan, Zeng Huang, Kripasindhu Sarkar, Danhang Tang, Di Qiu, Abhimitra Meka, Ruofei Du, Mingsong Dou, Sergio Orts-Escolano, et al. Learning personalized high quality volumetric head avatars from monocular rgb videos. In *CVPR*, pages 16890–16900, 2023. 1, 2, 3, 4, 5, 6, 7

[6] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W Sumner, and Markus Gross. High-Quality Passive Facial Performance Capture Using Anchor Frames. In *ACM SIGGRAPH 2011 Papers*, pages 1–10. 2011. 2

[7] Chen Cao, Tomas Simon, Jin Kyu Kim, Gabe Schwartz, Michael Zollhoefer, Shun-Suke Saito, Stephen Lombardi, Shih-En Wei, Danielle Belko, Shoou-I Yu, Yaser Sheikh, and Jason Saragih. Authentic volumetric avatars from a phone scan. *ACM TOG*, 2022. 2

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. 2

[9] Chuhan Chen, Matthew O'Toole, Gaurav Bharaj, and Pablo Garrido. Implicit neural head synthesis via controllable local deformation fields. In *CVPR*, pages 416–426, 2023. 2

[10] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 3

[11] Michail Christos Doukas, Mohammad Rami Koujan, Viktoriia Sharmanska, Anastasios Roussos, and Stefanos Zafeiriou. Head2head++: Deep Facial Attributes Re-Targeting. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 3(1):31–43, 2021. 3

[12] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3D Morphable Face Models—Past, Present, and Future. 39(5): 1–38, 2020. 2

[13] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *CVPR*, pages 8649–8658, 2021. 2, 5, 6

[14] Xuan Gao, Chenglai Zhong, Jun Xiang, Yang Hong, Yudong Guo, and Juyong Zhang. Reconstructing personalized semantic facial nerf models from monocular video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 41(6), 2022. 1, 2, 3, 4, 5, 6, 7

[15] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, Danhang Tang, Anastasia Tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Paul Debevec, and Shahram Izadi. The Relightables: Volumetric Performance Capture of Humans With Realistic Relighting. *ACM TOG*, 2019. 1, 2

[16] Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. NeuMan: Neural Human Radiance Field From a Single Video. 2022. 5

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):1–14, 2023. 3

[18] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep Video Portraits. *ACM TOG*, 37(4):1–14, 2018. 3

[19] Mohammad Rami Koujan, Michail Christos Doukas, Anastasios Roussos, and Stefanos Zafeiriou. Head2head: Video-Based Neural Head Synthesis. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 16–23. IEEE, IEEE, 2020. 3

[20] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a Model of Facial Shape and Expression From 4D Scans. *ACM TOG*, 36(6):194:1–194:17, 2017. 3

[21] Shanchuan Lin, Linjie Yang, Imran Saleemi, and Soumyadip Sengupta. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 238–247, 2022. 5

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[23] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG*, 40(4):1–13, 2021. 2

[24] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 5

[25] Abhimitra Meka, Rohit Pandey, Christian Haene, Sergio Orts-Escolano, Peter Barnum, Philip David-Son, Daniel Erickson, Yinda Zhang, Jonathan Taylor, Sofien Bouaziz, et al. Deep Relightable Textures: Volumetric Performance Capture With Neural Rendering. *ACM TOG*, 39(6):1–21, 2020. 2

[26] Abhimitra Meka, Rohit Pandey, Christian Haene, Sergio Orts-Escolano, Peter Barnum, Philip Davidson, Daniel Erickson, Yinda Zhang, Jonathan Taylor, Sofien Bouaziz,

Chloe Legendre, Wan-Chun Ma, Ryan Overbeck, Thabo Beeler, Paul Debevec, Shahram Izadi, Christian Theobalt, Christoph Rhemann, and Sean Fanello. Deep relightable textures - volumetric performance capture with neural rendering. 2020. 1

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2, 3, 4, 6, 8

[28] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. Holoportation: Virtual 3D Teleportation in Real-Time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 2016. 1, 2

[29] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields. pages 5865–5874, 2021. 5

[30] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 2

[31] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[32] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, pages 703–735. Wiley Online Library, 2022. 2

[33] Zach Waggoner. *My Avatar, My Self: Identity in Video Role-Playing Games*. McFarland, 2009. 1

[34] Lizhen Wang, Xiaochen Zhao, Jingxiang Sun, Yuxiang Zhang, Hongwen Zhang, Tao Yu, and Yebin Liu. Styleavatar: Real-time photo-realistic portrait avatar from a single video. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 3

[35] Yuelang Xu, Lizhen Wang, Xiaochen Zhao, Hongwen Zhang, and Yebin Liu. Avatarmav: Fast 3d head avatar reconstruction using motion-aware neural voxels. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 2

[36] Yuelang Xu, Hongwen Zhang, Lizhen Wang, Xiaochen Zhao, Han Huang, Guojun Qi, and Yebin Liu. Latentavatar: Learning latent expression code for expressive neural head avatar. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 3

[37] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2

[38] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features As a Perceptual Metric. pages 586–595, 2018. 5

[39] Xiaochen Zhao, Lizhen Wang, Jingxiang Sun, Hongwen Zhang, Jinli Suo, and Yebin Liu. Havatar: High-fidelity head avatar via facial model conditioned neural radiance field. *ACM TOG*, 43(1):1–16, 2023. 3

[40] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C Bühler, Xu Chen, Michael J Black, and Otmar Hilliges. Im avatar: Implicit morphable head avatars from videos. In *CVPR*, pages 13545–13555, 2022. 2

[41] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *CVPR*, pages 21057–21067, 2023. 1, 2, 3, 5, 6, 7

[42] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant volumetric head avatars. In *CVPR*, pages 4574–4584, 2023. 1, 2, 3, 4, 5, 6, 7

[43] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications. pages 523–550, 2018. 2