Published on CVPR 2023 Workshop
3DMV: Learning 3D with Multi-View Supervision

Multiface: A Dataset for Neural Face Rendering

Cheng-hsin Wuu*, Ningyuan Zheng*, Scott Ardisson, Rohan Bali, Danielle Belko, Eric Brockmeyer, Lucas Evans, Timothy Godisart, Hyowon Ha, Alexander Hypes, Xuhua Huang, Taylor Koska, Steven Krenn, Stephen Lombardi, Xiaomin Luo, Kevyn McPhail, Laura Millerschoen, Michal Perdoch, Mark Pitts, Alexander Richard, Jason Saragih, Junko Saragih, Takaaki Shiratori, Tomas Simon, Matt Stewart, Autumn Trimble, Xinshuo Weng, David Whitewolf, Chenglei Wu, Shoou-I Yu, Yaser Sheikh

Meta Reality Labs Research
{ecwuu, zhengningyuan}@meta.com

Abstract

Photorealistic avatars of human faces have come a long way in recent years, yet research along this area is limited by a lack of publicly available, high-quality datasets covering both, dense multi-view camera captures, and rich facial expressions of the captured subjects. In this work, we present Multiface, a new multi-view, high-resolution human face dataset collected from 13 identities at Reality Labs Research for neural face rendering. We introduce Mugsy, a large scale multi-camera apparatus to capture high-resolution synchronized videos of a facial performance. The goal of Multiface is to close the gap in accessibility to high quality data in the academic community and to enable research in VR telepresence. Along with the release of the dataset, we conduct ablation studies on the influence of different model architectures toward the model's interpolation capacity of novel viewpoint and expressions. With a conditional VAE model [8] serving as our baseline, we found that adding spatial bias, texture warp field, and residual connections improves performance on novel view synthesis. Our code and data is available at: https://github.com/facebookresearch/multiface.

1. Introduction

Photo-realistic human face rendering and reconstruction is essential to real-time telepresence technology that drives modern Virtual Reality applications. Since humans are social animals that have evolved to express and read emotions from subtle changes in facial expressions, tiny artifacts give rise to the uncanny valley that could hurt user experience. Nowadays, many modern 3D telepresence methods leverage deep learning models and neural rendering for high-fidelity reconstruction, and to tackle difficult problems such as novel view synthesis and view-dependent effects modeling [9, 12, 14, 15]. These approaches are usually data-hungry, and the design of a capture system and data collection pipeline directly determines the performance of those models. Pushing the boundaries in such photo-realistic human face models therefore requires a large dataset of high-resolution, multi-view facial images spanning a wide variety of expressions. We introduce such a dataset, collected by a high-end multi-view capturing system (*Mugsy*) that we built at Meta Reality Labs Research in Pittsburgh. Compared to existing face modeling datasets such as HUMBI [23] and FaceWarehouse [1], our Codec-Avatar dataset contains facial data of unprecedented quality, variation in facial expressions, and number of camera views. We capture 13 subjects with a great variety of high-fidelity facial expressions along with the geometry mesh tracked over the capturing time. For each subject, we have over a hundred facial expressions captured by multiple machine vision cameras synchronously at 4096×2668 resolution (about 11 megapixels). The capture system is illustrated in Figure 1.

We release the images of all 13 captured participants as well as tracked meshes and unwrapped textures, along with camera calibrations and audio data. Moreover, we provide code to train a Codec Avatar from scratch as well as pre-trained Codec Avatars for all identities.

The creation of data-driven avatars is challenging along three major axes: (1) novel view synthesis, as we could not have cameras placed everywhere (2) novel expression synthesis, as we could not ask the participants to enact all possible facial expressions during the capture, and (3) relighting, as it is impossible to capture every possible lighting configuration. In this report, we focus on the first two axes,

*Equal contribution.

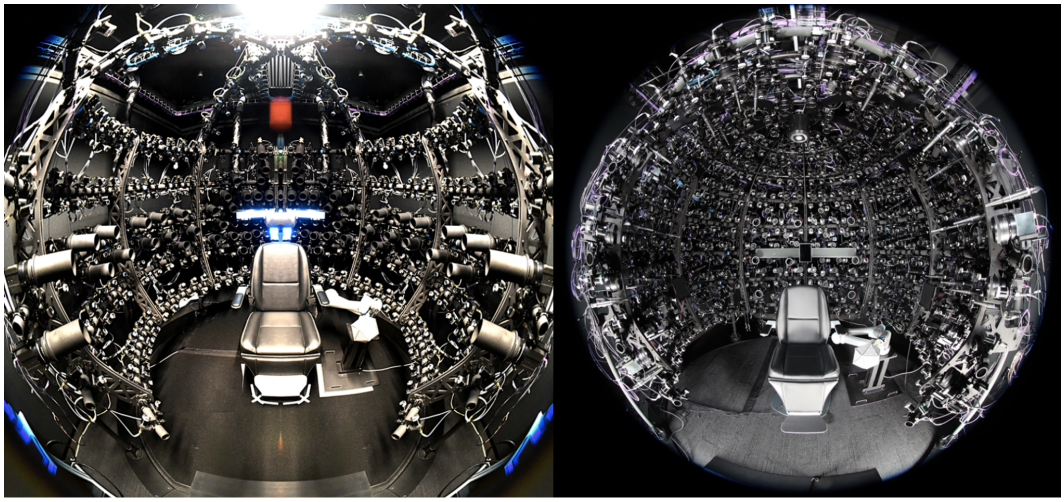


Figure 1. Mugsy v1 (left) and Mugsy v2 (right).

while relighting is beyond the scope of this work. We use a conditional VAE model [8] as a baseline, and evaluate the model's reconstruction quality with respect to different network architectures, which includes spatial biases, a texture warp field, and residual blocks. Empirically, we found that the baseline model benefits from these architectural modifications in interpolating novel views.

After addressing related work, we provide details on our capture system *Mugsy* in Section 3 and the capture script and process used to collect the dataset. In Section 4, we describe the model architectures and training pipeline for building the photo-realistic Codec Avatar. In Section 5, we present an ablation study of how different model architectures respond to synthesizing viewpoints and expressions.

2. Related Works

We briefly review two prior efforts in human face data collection, HUMBI [23] and FaceWarehouse [1], and compare these datasets with our dataset, see Table 1.

HUMBI [23] is a large-scale multi-view dataset designed to facilitate high resolution pose- and view-specific appearance of human body expressions (gaze, face, hand, body, and garment). The database uses a dense camera array composed of 107 synchronized cameras to capture 772 distinctive subjects doing diverse activities. The presence of HUMBI shows a new opportunity to build a versatile model that generates data-driven photo-realistic rendering for full body avatars. While HUMBI focuses on capturing multi-view images for the entire body, it does not provide high enough resolution for the images. This restriction poses a challenge on reconstructing the subtle changes of human facial expression, which we aim to overcome with our dataset.

Table 1. Comparison of multi-view datasets.

Dataset	Camera Resolution	# Expressions	# View
FaceWarehouse	640x480	20	1
HUMBI	1920x1080	20	32
Mugsy v1	2048x1334	65	40
Mugsy v2	2048x1334	118	150

FaceWarehouse [1] is a database of 3D facial expressions for visual computing applications. The database uses Kinect, an off-the-shelf RGBD camera, to capture and estimate facial geometry and texture of 150 subjects, each with 20 expressions. Compared with previous 3D facial databases, FaceWarehouse has richer collections of expressions for each person that enables depiction of most human facial actions. This dataset has potential on applications such as facial image manipulation, face component transfer, real-time performance-based facial image animation, and facial animation retargeting from video to image. However, the data in FaceWarehouse does not contain detailed facial geometries such as wrinkles due to the low precision in depth information provided from the capture apparatus. This insufficiency makes applications such as high-fidelity 3D facial reconstruction very challenging.

Our dataset, in contrast, provides the richness in facial expressions together with high-resolution images that enables us to model nuanced yet important subtleties in human faces up to the level of skin pores.

Existing works on Codec Avatars. In the past, a line of works emerged from our lab that is built on the data released in this dataset. Deep appearance models [8] are the first successful approach to building photorealistic, high quality

avatars but suffer from limitations in their expressiveness and controllability. To overcome these issues, in [2], the original deep appearance model has been replaced by a fully convolutional architecture that aims at increased modularity among different facial regions and thereby achieves higher expressiveness. Focusing on the importance of eye contact in human communication, [18] extends Codec Avatars with an explicit eye model. A limitation of all above approaches is their reliance on traditional mesh-based rendering, which falls short in its ability to accurately render thin structures, translucency, and biological motion. With the uprise of neural rendering, this reliance on mesh-based rendering has been largely overcome, and avatars have been shown to expose outstanding details and realistic modeling of difficult regions such as hair [9, 10, 13, 21, 22]. Audio-visual data has been leveraged in [16, 17], both in a mesh-based setup as well as in fully-textured avatar animation from audio and gaze. Finally, Pixel Codec Avatars [11] provide a lightweight model that can render photorealistic avatars on commodity hardware such as a Quest 2.

3. Dataset Creation

In this section, we detail how the dataset was created, starting with an overview of the dataset characteristics, followed by a capture system description, a description of the capture script participants went through, and the tracking pipeline used to process the captured data.

3.1. Dataset Overview

Our dataset consists of high quality recordings of the faces of 13 identities, each captured in a multi-view capture stage performing various facial expressions. An average of 12,200 (capture version 1) to 23,000 (capture version 2) frames per subject were captured at 30 fps. Each frame has roughly 40 (v1) or 150 (v2) different camera views under uniform illumination.

We provide the captured images from each camera view at a resolution of 2048 $\times$ 1334 pixels, tracked meshes including headposes, unwrapped textures at 1024 $\times$ 1024 pixels, metadata including intrinsic and extrinsic camera calibrations, and audio. Additionally, we release code to download the dataset and build a Codec Avatar using a deep appearance model [8]. All required code and dataset documentation will be publicly available online[1].

3.2. Capture Studio

In order to capture synchronized multi-view videos of a facial performance, we built a multi-video-camera capture dome called *Mugsy* (short from *Mugshooter*), see Figure 1.

The cameras are placed on the surface of a sphere with radius 1.2 meters. The cameras all point inward to the middle of the sphere, which is where the head of the participant is located. Figure 2 presents an overview of all the views. The sensors used are IMX253 with pixel size 3.45 μm. We capture at resolution 4096x2668 (11 megapixels). Shutter speed is at 2.222 ms. In order for the cameras to capture synchronously, they all listen to the rising edge of a single trigger. While the system is able to capture at 90fps, we only release data at 30fps and downsample the images to 2048x1334 to limit the total dataset size. For lights, we use point light sources that are pointing towards the center of the sphere to illuminate the face of the participant. The lights have diffusers installed to reduce specular highlights on the person's face and better approximate uniform lighting. All cameras are jointly calibrated using a 3D calibration target [4] mounted on a robot arm. The calibration process is based on corner detection, intrinsics calibration, and then a final bundle adjustment. Intrinsics and extrinsics of each camera and for each participant are provided as part of the dataset.

We release ten captures from the original Mugsy (v1) and three captures obtained with an extended version featuring a significantly larger amount of cameras (Mugsy v2), see Table 2 for details. Note that the number of cameras in each capture may be less than the number of cameras shown in the table as individual cameras might fail during a capture.

3.3. Capture Script

The goal of the face captures is to cover the full range of facial expressions such that a neural avatar like a deep appearance model [8] can learn to interpolate from a range of captured expressions to all possible facial expressions. To this end, we design a capture script that captures a range of expressions, gaze directions, and 50 phonetically balanced sentences, see Table 3. The ten captures from Mugsy v1 follow a script where the expression portion is focused on peak expressions as in Figure 3, which aim to capture motion in different regions of the face independently. The three captures from Mugsy v2 follow a slightly modified script, where the expression portion is focused on full-face range of motion tasks, and the gaze portion has been simplified.

3.4. Data Processing: The Tracking Pipeline

In order to obtain meshes and unwrapped textures from the raw images, we run the captured data through a sophisticated tracking pipeline. Note that we release the resulting meshes and textures, so rebuilding the tracking pipeline is not required for users of this dataset. The pipeline follows several steps as illustrated in Figure 4. First, we run a modified version of parallel Patchmatch [3] on each frame to get a dense 3D mesh reconstruction. Note that dense meshes in different frames do not share the same topology. Next, we detect 2D keypoints on the face. Then, we run sequential tracking with model-free mesh tracking [20] with images,

[1] https://github.com/facebookresearch/multiface

Table 2. Specifications for each generation of Mugsy.

Mugsy	# Cameras	Camera resolution	Camera lens	FPS	# Lights
v1	40 color + 50 monochrome	4096x2668	35mm * 2, 85mm * 12, Remaining are 50 mm	30/90	350
v2	160 color	4096x2668	All 35 mm	30/90	450

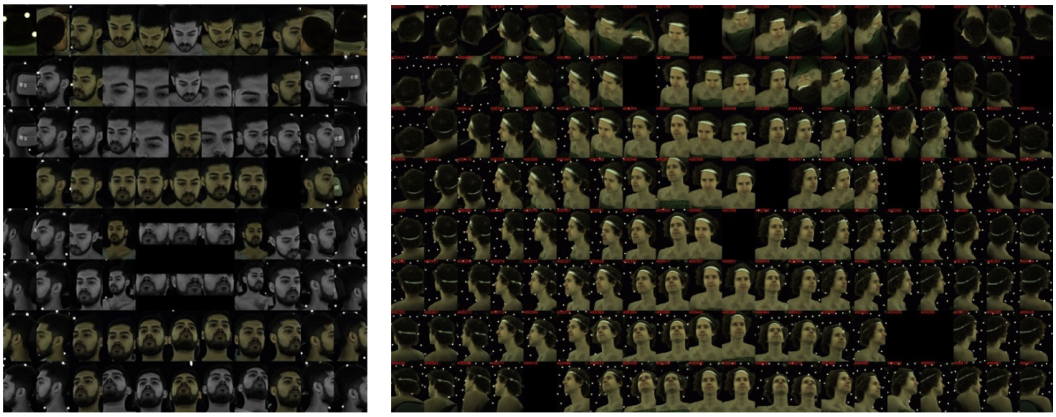


Figure 2. Left: camera views for Mugsy v1. Right: camera views for Mugsy v2.

keypoints, and the dense 3D mesh reconstructions as input to get corresponding tracked meshes. Due to the computational cost of these steps, we run this sequential tracking pipeline only on a subset of the captured data, *i.e.*, the expressions and the gaze portion, but not the sentences. Given these tracked results, we generate training data for personalized keypoint detection. The personalized keypoints are not just constrained to landmarks where a human annotator can consistently annotate, but also locations on the cheek and forehead, which is hard for a human to annotate consistently, but could be annotated accurately by model-free mesh tracking. The training data is used to train a personalized keypoint detector, which is then used as an initialization of a PCA model-based mesh tracking method. The advantage of using the personalized keypoints and PCA model-based tracking is that sequential tracking is no longer required, and all frames can be tracked in parallel, thus reducing the computational cost significantly and allowing us to process the complete capture efficiently. Finally, once we have the tracked meshes and an image from each view, we unwrap the texture for that specific view to obtain all necessary data for codec avatar generation.

3.5. Dataset Summary

In summary, for each of the 13 captured subjects, we provide the following data.

Raw Images. Raw images are directly captured from 40 (v1) to 160 (v2) multi-view cameras at the rate of 30 fps and are released at the resolution of 2048 $\times$ 1334. Raw images can be used as ground truth to compute the screen loss with predicted rendered images from the model.

Figure 3. The 65 peak expressions used in the v1 script. Participants are asked to make their best effort for expressions they may not be able to do, e.g., raise only left or right eyebrow.

Unwrapped Textures. Unwrapped textures are provided at a resolution of 1024 $\times$ 1024, and are generated by unwrapping the raw images from the geometry. We wrap each mesh triangle to the corresponding UV texture triangle using barycentric interpolation. Each camera and each frame has its unique view-dependent unwrapped texture.

Tracked Meshes. Meshes are tracked per frame and stored in *.obj* format with same topology. Each mesh consists of 7,306 vertices, with no vertices inside eyes or mouth. By projecting with the provided camera calibrations and headposes, meshes can be aligned with raw images.

Headposes. A headpose is a 3 $\times$ 4 matrix consisting of rotation and translation that represents the rigid body transformation of the head mesh at each frame.

Table 3. Comparison between v1 and v2 capture scripts.

Script	Expressions	Gaze	Sentences
v1	65 peak expressions as shown in Fig. 3 Participants go from neutral expression to peak to neutral. Only data from peak to neutral is processed and released. 1 single range-of-motion segment: ROM07	Participants look at 25 fixed markers without turning their head. Participants will look at the leds with normal eyes, wide eyes, squinty eyes, and small head rotations.	50 phonetically balanced sentences
v2	2 peak expressions: neutral and eyes closed mouth lightly open. 18 range-of-motion segments covering 118 expressions over the entire face.	Participants look at 9 cameras without turning their head. When looking at each camera, look at it with normal eyes, wide eyes, squinty eyes, blink, and wink. Additionally, look at a camera and do 5 head rotations.	50 phonetically balanced sentences

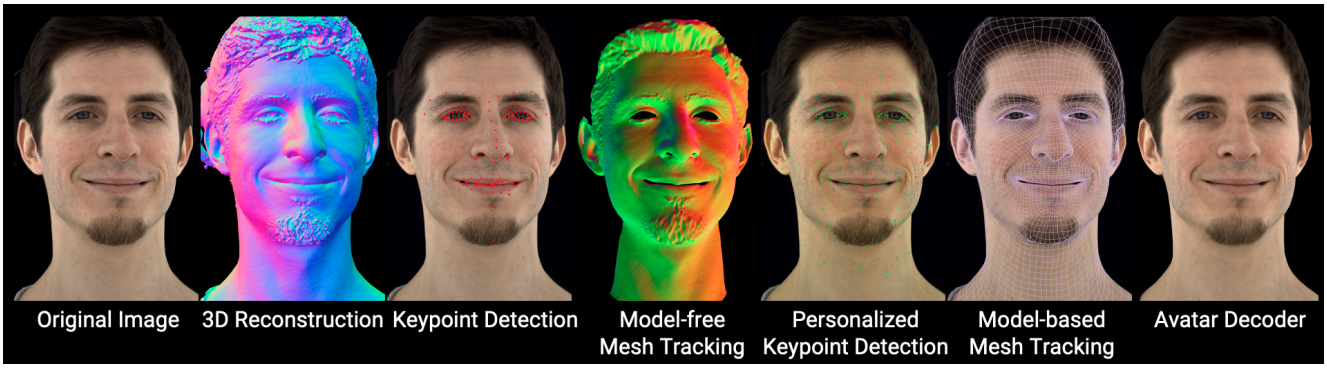


Figure 4. Snapshot of intermediate results from different parts of the pipeline.

Audio. While most captured expressions are silent, each participant was asked to read 50 phonetically balanced sentences. We provide audio data for these 50 sentences of each participant.

Metadata. The following metadata is provided as well:

- *camera calibrations:* we provide each camera's intrinsic and extrinsic matrix.
- *frame list:* a list of all frames captured by Mugsy, each line consists of segment name and frame index.
- *texture mean:* the mean of the textures across all frames and all cameras.
- *texture variance:* the variance of the textures across all frames and all cameras.
- *vertex mean:* the mean of the vertices of the meshes across all frames and all cameras.
- *vertex variance:* the variance of the vertices of the meshes across all frames and all cameras.

4. Technical Details

In this section, we demonstrate how to use the dataset and the provided scripts for training. For detailed instructions how to train a codec avatar using our codebase, see the Github repository.

Table 4. Mugsy and script version, # (color) cameras, and # frames for each released capture.

Capture ID (Mugsy/Script Version)	# Cameras	# Frames
m–20171024–0000–002757580–GHS (v1)	39	9791
m–20180105–0000–002539136–GHS (v1)	39	9903
m–20180226–0000–6674443–GHS (v1)	39	12381
m–20180227–0000–6795937–GHS (v1)	38	13478
m–20180406–0000–8870559–GHS (v1)	40	14914
m–20180418–0000–2183941–GHS (v1)	40	15115
m–20180426–0000–002643814–GHS (v1)	40	10258
m–20180510–0000–5372021–GHS (v1)	40	12619
m–20180927–0000–7889059–GHS (v1)	34	13541
m–20181017–0000–002914589–GHS (v1)	34	10179
m–20190529–1004–5067077–GHS (v2)	146	22928
m–20190529–1300–002421669–GHS (v2)	150	24375
m–20190828–1318–002645310–GHS (v2)	147	21399

4.1. Model

Our model greatly resembles the deep appearance model [8], which, at its core, is a variational autoencoder (VAE) that takes meshes and average texture as input and decodes the view-dependent textures for rendering, see Figure 5. We follow the original algorithm of [8] with some minor exceptions that will be outlined in the following.

As input, the model consumes the average texture from

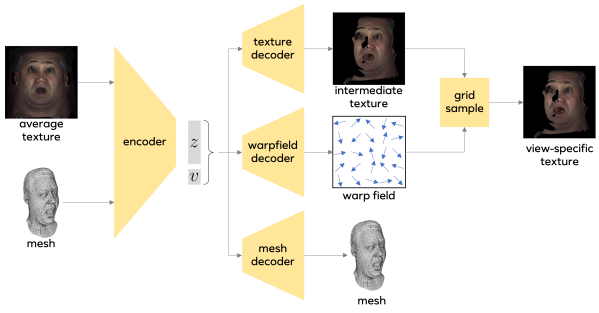


Figure 5. The model consumes, as input, a tracked mesh and an average texture over all camera views and maps these inputs to a latent code, from which a view-dependent texture and the tracked mesh can be decoded.

all camera views for a given training frame as well as the tracked mesh. Using a neural encoder, these inputs are mapped to a 256-dimensional KL-regularized latent space. Note that the latent space is view-independent as the inputs to the encoder are view-agnostic. A neural decoder consumes such a view-independent latent representation and a view vector and generates the texture for this specific view as well as the mesh.

The texture encoder consists of 8 convolutional layers, each with kernel size 4 and stride 2, that downsamples the input texture from a resolution of 1024x1024 to a 4x4 feature map. The mesh is encoded with a multi-layer perceptron (MLP) and encoding from texture encoder and mesh encoder are combined into a single 256-dimensional latent vector using a fully connected layer. On the decoder side, the view information is fed into an MLP and the view-feature is concatenated with the latent code. Thus, the texture decoder can be conditioned on this view information and models view-dependent effects in texture space. We explore several different architectures to investigate their generalizing capacity on novel expressions and camera views.

Color Correction. Since different cameras could have different color space, we optimize color correction parameters for each camera. Color correction is performed on the output texture by scaling and adding a bias to each RGB channel. The scaling factors and biases are initialized to 1 and 0, respectively. We fix the color correction parameters of one camera as an anchor and train the other parameters as a part of the model. Applying color correction is necessary, otherwise the reconstruction error will be dominated by the global color difference instead of exact colors of the pixels.

Spatial Bias. For convolutional layers in the decoder for upsampling, instead of adding the same bias value per channel in the feature map, we add a bias tensor that has the same shape as the feature map, meaning that each spatial location has its own bias value. In this way, the model is able to capture more position-specific details in the texture, such as wrinkles and lips.

Warp Field. We can also decode a warp field from the latent space and bilinearly sample the output texture with the warp field. Conceptually, texture generation can be decomposed into two steps: a synthesized texture on a deformation-free template followed by a deformation field that introduces shape variability. Denote $T(p)$ as the value of the synthesized texture at coordinate $p = (x, y)$. Denote $W(p)$ as the estimated deformation field at location p. Then, the observed image, $I(p)$, can be reconstructed as follows: $I(p) = T(W(p))$, namely the image appearance at position p is obtained by looking up the synthesized appearance at position $W(p)$. We obtain a warp field in the same way as deformable autoencoders [19]: by integrating both vertically and horizontally on the generated warping grid to avoid flipping of relative pixel positions.

Residual Connection. We insert residual layers [5] into our network to make it deeper. We investigate whether this increase in model capacity would make it generalize better.

4.2. Training Pipeline

In [8], the model was trained with an ℓ_2-loss on predicted mesh vertices and textures. We present a set of experiments that diverge from this training strategy by optimizing the screen space loss of the predicted avatars directly against the ground truth images. To propagate gradients from screen space to the predicted textures and geometries, we use Nvdiffrast [7] as the differentiable rendering engine.

More formally, given a ground truth image $I(v)$ captured from a camera at viewpoint v, the loss can be computed by rendering the predicted texture $\hat{T}$ and the predicted geometry $\hat{G}$ from viewpoint v using the differentiable rendering function R, and comparing it to the respective ground truth image $I(v)$,

$$L = \|R(v, \hat{T}, \hat{G}) - I(v)\|^2. \quad (1)$$

However, naively computing an ℓ_2-loss from the 3D rendered avatar in screen space and the ground truth images is not reasonable: ground truth images include background that must not be learned by the model. Moreover, humans are more sensitive to changes in around the eyes and mouth. To mitigate these two issues, we apply a *foreground mask* F (defined in texture space) calculated from non-background pixels in image space and assign a higher weight to eyes and mouth regions during training than to the remaining face regions using a manually created *texture weight mask* M. We therefore render both the predicted geometry $\hat{G}$ and the predicted texture $\hat{T}$ into screen space, but also the weight mask. The resulting loss then is defined as

$$L = \|R(v, M, \hat{G}) \odot R(v, \hat{T}, \hat{G}) \odot F - I(v) \odot F\|^2, \quad (2)$$

where $\odot$ denotes element-wise multiplication.

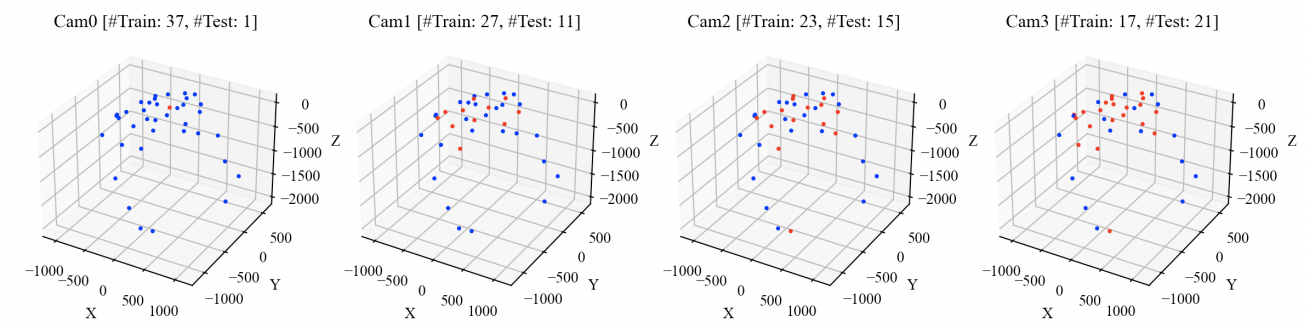


Figure 6. Sets of training and testing camera splits used in ablation study. Red dots represents the position of testing cameras and blue dots represents the position of training cameras. Note that for better visibility of the camera sets, we rendered the plots such that the z-axis is the one pointing away from the identity's face.

We add an explicit loss between predicted and (tracked) ground truth geometry to enhance learning the right geometric shape of the face during training as well as the KL loss for the VAE. The combined loss term is

$$\begin{aligned} L =& \lambda_1 \cdot \|R(v, M, \hat{G}) \odot R(v, \hat{T}, \hat{G}) \odot F - I(v) \odot F\|^2 \\ &+ \lambda_2 \cdot \|\hat{G} - G\|^2 \\ &+ \lambda_3 \cdot \mathrm{KL}(\mathcal{N}(\mu_z, \sigma_z) || \mathcal{N}(0, I)), \end{aligned} \tag{3}$$

with μ_z and σ_z being the predicted mean and variance of the latent distribution. Here the images and textures are normalized by per-pixel mean and variance during training. For faster convergence and unequal learning rate, we multiply the output mean of the encoder by 0.1 and the log standard deviation by 0.01. We use Adam [6] as optimizer and perform 200K iterations for all the experiments. We set $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 0.01$.

5. Experiments

In the following, we evaluate the changes to the original model from [8] that have been suggested in Section 4.1.

5.1. Experiment Setting

The ablation study is performed on a single identity, m–20180227–0000–6795937–GHS, and with varying training and test camera views. To evaluate the effect of the number of camera views available during training, we run each experiment with four different settings: (1) training on 37 cameras, (2) training on 27 cameras, (3) training on 23 cameras, and (4) training on 17 cameras. Figure 6 illustrates the location of the train cameras for these four settings. In total, training takes around one day for each avatar using a P3.16x instance with eight Nvidia V100 GPUs on AWS.

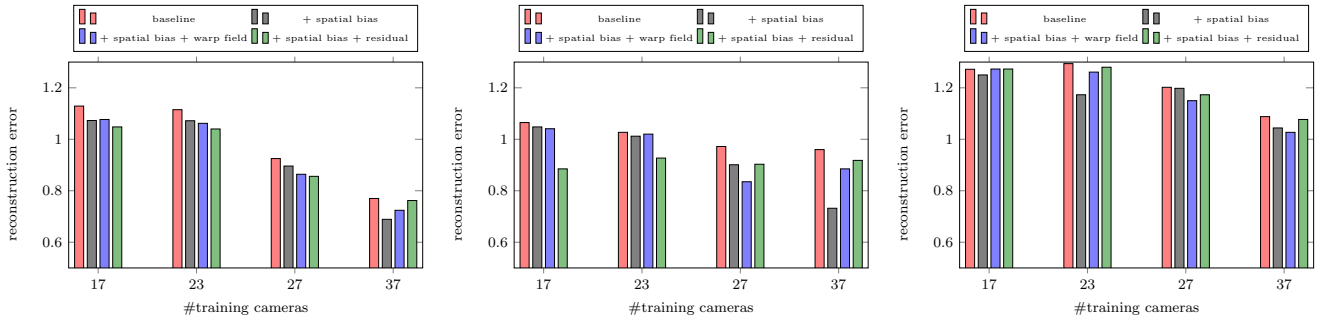
We conduct ablation studies for four different model architectures: (1) the model from [8] without the spatial bias, (2) the original model from [8] including spatial bias, (3) the model plus spatial bias and warp field, and (4) the model with spatial bias and residual connections, as described in Section 4.1.

Building photorealistic 3D facial avatars requires models to accurately produce novel views on the avatar (novel view synthesis), and to generalize well to facial expressions that are unseen in the training set (novel expression synthesis). For novel view synthesis, we evaluate the reconstruction error on held out cameras (red dots in Figure 6) and expressions seen during training, and for novel expression synthesis we evaluate the reconstruction error on camera views used during training (blue dots in Figure 6) but with held out facial expressions. We also examine both properties jointly by evaluating on held out cameras and held out expressions.

For fair comparison, we train color correction parameters for unseen testing cameras for two epochs on the validation data before evaluation. We also fine-tune the encoder on the testing expression as we would like to test the capacity of the conditional decoder independent of the encoder. To enable the encoder to generate the latent code that is specialized to the given dataset, we therefore train the encoder on the validation data while freezing the decoder weights on given cameras for 10 epochs before evaluation.

5.2. Results and Analysis

Novel View Synthesis. We first evaluate the model with respect to its ability to synthesize novel views. In Figure 7a, we plot the screen error measured on held out cameras over the number of available training cameras. The expressions we evaluate on are seen during training but the testing cameras are at unseen positions. The higher the number of available training cameras, the lower the reconstruction error. Spatial biases in the network are crucial for high accuracy in novel view synthesis as they encode view-independent texture information, compare the baseline without spatial bias (red bar) to all other architectures that have spatial biases. An increased number of layers achieved through the use of



(a) novel view synthesis (b) novel expression synthesis (c) novel view + novel expression

	(a) novel view synthesis				(b) novel expression synthesis				(c) novel view+expression synthesis			
# training cameras	17	23	27	37	17	23	27	37	17	23	27	37
baseline	1.129	1.115	0.925	0.770	1.065	1.027	0.972	0.960	1.272	1.294	1.202	1.088
+ spatial bias	1.073	1.072	0.896	0.689	1.048	1.012	0.901	0.732	1.250	1.173	1.198	1.044
+ spatial bias + warp-field	1.077	1.062	0.864	0.724	1.041	1.020	0.835	0.885	1.273	1.261	1.150	1.027
+ spatial bias + residual	1.048	1.040	0.856	0.762	0.885	0.927	0.903	0.918	1.273	1.280	1.173	1.077

Figure 7. Reconstruction errors for four architectures trained on 17, 23, 27, and 37 camera views. We compare system performance for novel view synthesis, novel expression synthesis, and joint synthesis of novel expressions in novel views. The reconstruction errors refer to pixel intensity difference.

residual connections is particularly important for a lower number of training cameras (green bar).

Novel Expression Synthesis. To evaluate the model's ability to generate unseen facial expressions, we therefore evaluate on seen views (*i.e.*, training cameras) only, see Figure 7b. As before, a lack of spatial biases leads to a significant degradation of reconstruction quality compared to other architectures that use spatial biases. While the baseline model plus spatial bias performs best when dense training views are available, it quickly deteriorates for fewer available training cameras. The deeper model with residual connections shows the most consistent performance, being at a rather low reconstruction error independent of the number of available training cameras.

Joint View- and Expression Synthesis. Last, we combine the two previous settings and evaluate the performance of our four architectures on unseen facial expressions and novel camera views simultaneously, see Figure 7c. As before, the larger the number of available training camera views, the smaller the reconstruction error. Note, however, that each individual task alone is easier to solve than the joint task: the reconstruction errors for novel view synthesis and novel expression synthesis alone are significantly lower than the reconstruction errors for the joint task of novel view- and expression synthesis.

Qualitative Results. Figure 8 shows predicted frames after training of a warp-field model. The top row shows ground-truth and the bottom shows rendered image using predicted mesh and texture. Although the reconstruction generally looks good, the model struggles to predict high frequency details such as teeth and eye-lashes.

Figure 8. Reconstruction result of the warp-field model using 37 camera views during training. Top row is ground-truth, bottom row is the 3D reconstruction generated by the model.

6. Conclusion

We release a large-scale multi-view codec-avatar dataset for neural face rendering, along with training, evaluation, and visualization code and pretrained models for 13 identities. Besides the dataset, to understand how different model architectures respond to interpolating on unseen viewpoint and expression, we conduct an ablation study and identify that the baseline VAE model benefits from adding spatial bias, texture warp field and residual connections. We hope that this dataset will push the limit of facial reconstruction further and facilitate future research of VR telepresence.

References

[1] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014. 1, 2

[2] Hang Chu, Shugao Ma, Fernando De la Torre, Sanja Fidler, and Yaser Sheikh. Expressive telepresence via modular codec avatars. In *European Conference on Computer Vision*, pages 330–345. Springer, 2020. 3

[3] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, 2015. 3

[4] Hyowon Ha, Michal Perdoch, Hatem Alismail, In So Kweon, and Yaser Sheikh. Deltille grids for geometric camera calibration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5344–5352, 2017. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 6

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 7

[7] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 6

[8] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 37(4), jul 2018. 1, 2, 3, 5, 6, 7

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes. *ACM Transactions on Graphics*, 38(4):1–14, Aug 2019. 1, 3

[10] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. 3

[11] Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De La Torre, and Yaser Sheikh. Pixel codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 64–73, 2021. 3

[12] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. 1

[13] Giljoo Nam, Chenglei Wu, Min H Kim, and Yaser Sheikh. Strand-accurate multi-view hair capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 155–164, 2019. 3

[14] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *CoRR*, abs/2011.12948, 2020. 1

[15] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2021. 1

[16] Alexander Richard, Colin Lea, Shugao Ma, Jurgen Gall, Fernando De la Torre, and Yaser Sheikh. Audio-and gaze-driven facial animation of codec avatars. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 41–50, 2021. 3

[17] Alexander Richard, Michael Zollhöfer, Yandong Wen, Fernando De la Torre, and Yaser Sheikh. Meshtalk: 3d face animation from speech using cross-modality disentanglement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1173–1182, 2021. 3

[18] Gabriel Schwartz, Shih-En Wei, Te-Li Wang, Stephen Lombardi, Tomas Simon, Jason Saragih, and Yaser Sheikh. The eyes have it: An integrated eye and face model for photorealistic facial animation. *ACM Transactions on Graphics (TOG)*, 39(4):91–1, 2020. 3

[19] Zhixin Shu, Mihir Sahasrabudhe, Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance, 2018. 6

[20] Breannan Smith, Chenglei Wu, He Wen, Patrick Peluse, Yaser Sheikh, Jessica K. Hodgins, and Takaaki Shiratori. Constraining dense hand surface tracking with elasticity. *ACM Trans. Graph.*, 39(6), nov 2020. 3

[21] Tiancheng Sun, Giljoo Nam, Carlos Aliaga, Christophe Hery, and Ravi Ramamoorthi. Human hair inverse rendering using multi-view photometric data. 2021. 3

[22] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhofer. Learning compositional radiance fields of dynamic human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2021. 3

[23] Zhixuan Yu, Jae Shin Yoon, In Kyu Lee, Prashanth Venkatesh, Jaesik Park, Jihun Yu, and Hyun Soo Park. Humbi: A large multiview dataset of human body expressions, 2020. 1, 2