Deep Appearance Models for Face Rendering

STEPHEN LOMBARDI, Facebook Reality Labs
JASON SARAGIH, Facebook Reality Labs
TOMAS SIMON, Facebook Reality Labs
YASER SHEIKH, Facebook Reality Labs

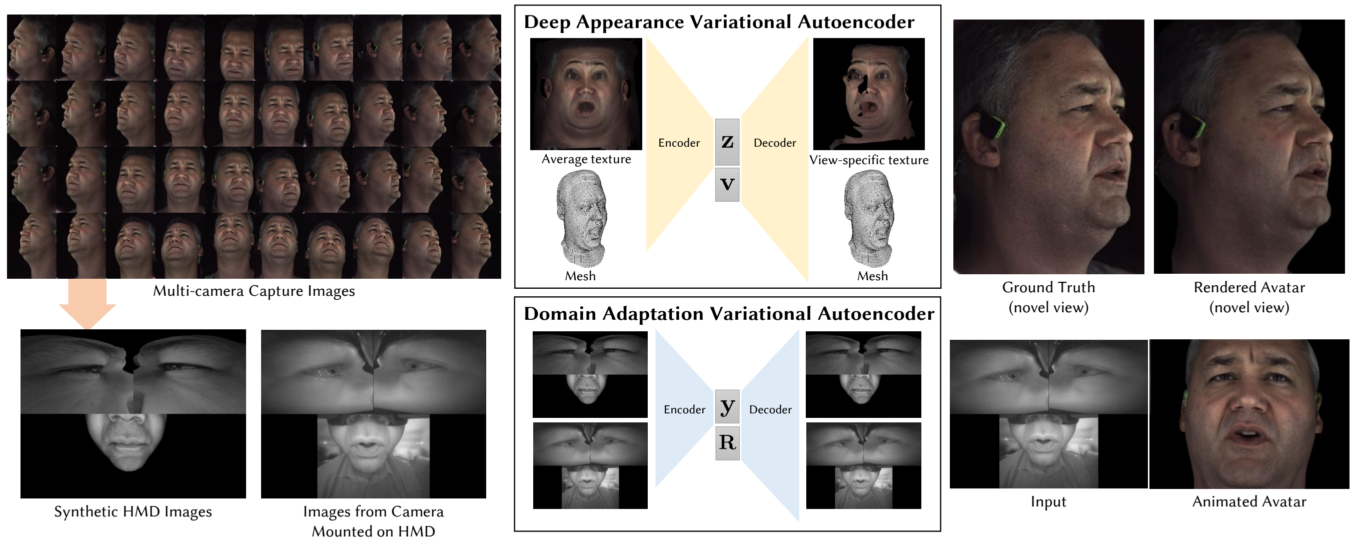
Multi-camera Capture Images
Synthetic HMD Images
Images from Camera Mounted on HMD
Deep Appearance Variational Autoencoder
Average texture
Mesh
Encoder
z
v
Decoder
View-specific texture
Mesh
Domain Adaptation Variational Autoencoder
Encoder
y
R
Decoder
Ground Truth (novel view)
Rendered Avatar (novel view)
Input
Animated Avatar

Fig. 1. Our model jointly encodes and decodes geometry and view-dependent appearance into a latent code **z**, from data captured from a multi-camera rig, enabling highly realistic data-driven facial rendering. We use this rich data to drive our avatars from cameras mounted on a head-mounted display (HMD). We do this by creating synthetic HMD images through image-based rendering, and using another variational autoencoder to learn a common representation **y** of real and synthetic HMD images. We then regress from **y** to the latent rendering code **z** and decode into mesh and texture to render. Our method enables high-fidelity social interaction in virtual reality.

We introduce a deep appearance model for rendering the human face. Inspired by Active Appearance Models, we develop a data-driven rendering pipeline that learns a joint representation of facial geometry and appearance from a multiview capture setup. Vertex positions and view-specific textures are modeled using a deep variational autoencoder that captures complex non-linear effects while producing a smooth and compact latent representation. View-specific texture enables the modeling of view-dependent effects such as specularity. In addition, it can also correct for imperfect geometry stemming from biased or low resolution estimates. This is a significant departure from the traditional graphics pipeline, which requires highly accurate geometry as well as all elements of the shading model to achieve realism through physically-inspired light transport. Acquiring such a high level of accuracy is difficult in practice, especially for complex and intricate parts of the face, such as eyelashes and the oral cavity. These are handled naturally by our approach, which does not rely on precise estimates of geometry. Instead, the shading model accommodates deficiencies in geometry though the flexibility afforded by the neural network employed. At inference time, we condition the decoding network on the viewpoint of the camera in order to generate the appropriate texture for rendering. The resulting system can be implemented simply using existing rendering engines through dynamic textures with flat lighting. This representation, together with a novel unsupervised technique for mapping images to facial states, results in a system that is naturally suited to real-time interactive settings such as Virtual Reality (VR).

CCS Concepts: • **Computing methodologies → Image-based rendering**; *Neural networks*; *Virtual reality*; *Mesh models*;

Additional Key Words and Phrases: Appearance Models, Image-based Rendering, Deep Appearance Models, Face Rendering

ACM Reference Format:
Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep Appearance Models for Face Rendering. *ACM Trans. Graph.* 37, 4, Article 1 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201401

Authors' addresses: Stephen Lombardi, Facebook Reality Labs, Pittsburgh, PA, stephen.lombardi@fb.com; Jason Saragih, Facebook Reality Labs, Pittsburgh, PA, jason.saragih@fb.com; Tomas Simon, Facebook Reality Labs, Pittsburgh, PA, tomas.simon@fb.com; Yaser Sheikh, Facebook Reality Labs, Pittsburgh, PA, yaser.sheikh@fb.com.

1 INTRODUCTION

With the advent of modern Virtual Reality (VR) headsets, there is a need for improved real-time computer graphics models for enhanced immersion. Traditional computer graphics techniques are capable of creating highly realistic renders of static scenes but at high computational cost. These models also depend on physically accurate estimates of geometry and shading model components. When high precision estimates are difficult to acquire, degradation in perceptual quality can be pronounced and difficult to control and anticipate. Finally, photo-realistic rendering of dynamic scenes in real time remains a challenge.

Rendering the human face is particularly challenging. Humans are social animals that have evolved to express and read emotions in each other's faces [Ekman 1980]. As a result, humans are extremely sensitive to rendering artifacts, which gives rise to the well-known uncanny valley in photo-realistic facial rendering and animation. The source of these artifacts can be attributed to deficiencies, both in the rendering model as well as its parameters. This problem is exacerbated in real-time applications, such as in VR, where limited compute budget necessitates approximations in the light-transport models used to render the face. Common examples here include: material properties of the face that are complex and time-consuming to simulate; fine geometric structures, such as eyelashes, pores, and vellus hair that are difficult to model; and subtle dynamics from motion, particularly during speech. Although compelling synthetic renderings of faces have been demonstrated in the movie industry, these models require significant manual cleanup [Lewis et al. 2014], and often a frame-by-frame adjustment is employed to alleviate deficiencies of the underlying model. This manual process makes them unsuitable for real-time applications.

The primary challenge posed by the traditional graphics pipeline stems form its physics-inspired model of light transport. For a specific known object class, like a human face, it is possible to circumvent this generic representation, and instead directly address the problem of generating images of the object using image statistics from a set of examples. This is the approach taken in Active Appearance Models (AAMs) [Cootes et al. 2001], which synthesize images of the face by employing a joint linear model of both texture and geometry (typically a small set of 2D points), learned from a collection of face images. Through a process coined analysis-by-synthesis, AAMs were originally designed to infer the underlying geometry of the face in unseen images by optimizing the parameters of the linear model so that the synthesized result best matches the target image. Thus, the most important property of AAMs is their ability to accurately synthesize face images while being sufficiently constrained to only generate plausible faces. Although AAMs have been largely supplanted by direct regression methods for geometry registration problems [Kazemi and Sullivan 2014; Xiong and la Torre Frade 2013], their synthesis capabilities are of interest due to two factors. First, they can generate highly realistic face images from sparse 2D geometry. This is in contrast to physics-inspired face rendering that requires accurate, high-resolution geometry and material properties. Second, the perceptual quality of AAM synthesis degrades gracefully. This factor is instrumental in a number of perceptual experiments that rely on the uncanny-valley being overcome [Boker et al. 2011; Cassidy et al. 2016].

Inspired by the AAM, in this work we develop a data-driven representation of the face that jointly models variations in sparse *3D* geometry and *view-dependent* texture. Departing from the linear models employed in AAMs, we use a deep conditional variational autoencoder [Kingma and Welling 2013] (CVAE) to learn the latent embedding of facial states and decode them into rendering elements (geometry and texture). A key component of our approach is the view-dependent texture synthesis, which can account for limitations posed by sparse geometry as well as complex nonlinear shading effects such as specularity. We explicitly factor out viewpoint from the latent representation, as it is extrinsic to the facial performance, and instead condition the decoder on the direction from which the model is viewed. To learn a deep appearance model of the face, we constructed a multiview capture apparatus with 40 cameras pointed at the front hemisphere of the face. A coarse geometric template is registered to all frames in a performance and it, along with the unwarped textures of each camera, constitute the data used to train the CVAE. We investigated the role of geometric precision and viewpoint granularity and found that, although direct deep image generation techniques such as those of Radford et al. [2015], Hou et al. [2017], or Kulkarni et al. [2015] can faithfully reconstruct data, extrapolation to novel viewpoints is greatly enhanced by separating the coarse graphics warp (via a triangle mesh) from appearance, confirming the basic premise of the parameterization.

Our proposed approach is well suited for visualizing faces in VR since; a) sparse geometry and dynamic texture are basic building blocks of modern real-time rendering engines, and b) modern VR hardware necessarily estimates viewing directions in real-time. However, in order to enable interaction using deep appearance models in VR, the user's facial state needs to be estimated from a collection of sensors that typically have extreme and incomplete views of the face. An example of this is shown in the bottom left of Figure 1. This problem is further complicated in cases where the sensors measure a modality that differs from that used to build the model. A common example is the use of IR cameras, which defeats naive matching with images captured in the visible spectrum due to pronounced sub-surface scattering in IR. To address this problem, we leverage the property of CVAE, where weight sharing across similar modalities tends to preserve semantics. Specifically, we found that learning a common CVAE over headset images and re-rendered versions of the multiview capture images allows us to correspond the two modalities through their common latent representation. This, in turn, implies correspondence from headset images to latent codes of the deep appearance model, over which we can learn a regression. Coupled with the deep appearance model, this approach allows the creation of a completely personalized end-to-end system where a person's facial state is encoded with the tracking VAE encoder and decoded with the rendering VAE decoder without the need for manually defined correspondences between the two domains.

This paper makes two major contributions to the problem of real-time facial rendering and animation:

- A formulation of deep appearance models that can generate highly realistic renderings of the human face. The approach does not rely on high-precision rendering elements, can be built completely automatically without requiring any manual cleanup, and is well suited to real-time interactive settings such as VR.
- A semi-supervised approach for relating the deep appearance model latent codes to images captured using sensors with

drastically differing viewpoint and modality. The approach does not rely on any manually defined correspondence between the two modalities but can learn a direct mapping from images to latent codes that can be evaluated in real-time.

In §2 we cover related work, and describe our capture apparatus and data preparation in §3. The method for building the deep appearance models is described in §4, and the technique for driving it from headset mounted sensors in §5. Qualitative and quantitative results from investigating various design choices in our approach are presented in §6. We conclude in §7 with a discussion and directions of future work.

2 RELATED WORK

Active Appearance Models (AAMs) [Cootes et al. 2001; Edwards et al. 1998; Matthews and Baker 2004; Tzimiropoulos et al. 2013] and 3D morphable models (3DMM) [Blanz and Vetter 1999; Knothe et al. 2011] have been used as an effective way to register faces in images through analysis-by-synthesis using a low dimensional linear subspace of both shape and appearance. Our deep appearance model can be a seen as a re-purposing of the generator component of these statistical models for rendering highly realistic faces. To perform this task effectively, there are two major modifications that were necessary over the classical AAM and 3DMM approaches; the use of deep neural networks in-place of linear generative functions, and a system for view-conditioning to specialize the generated image to the viewpoint of the camera.

Recently, there has been a great deal of work using deep networks to model human faces in images. Radford et al. [2015] use Generative Adversarial Networks (GANs) to learn representations of the face and learn how movements in this latent space can change attributes of the output image (e.g., wearing eyeglasses or not). Hou et al. [2017] use a variational autoencoder to perform a similar task. Kulkarni et al. [2015] attempt to separate semantically meaningful characteristics (e.g., pose, lighting, shape) from a database of rendered faces semi-automatically. All these methods operate only in the image domain and are restricted to small image sizes (256×256 or less). By contrast, we learn to generate view-specific texture maps of how the face changes due to facial expression and viewing direction. This greatly simplifies the learning problem, achieving high-resolution realistic output as a result.

There has also been work on automatically creating image-based avatars using a geometric facial model [Hu et al. 2017; Ichim et al. 2015; Thies et al. 2018]. Cao et al. [2016] propose an image-based avatar from 32 images captured from a webcam. To render the face, they construct a specially crafted function that blends the set of captured images based on the current facial expression and surface normals, which captures the dependence of both expression and viewpoint on facial appearance. Casas et al. [2016] propose a method for rapidly creating blendshapes from RGB-D data, where the face is later rendered by taking a linear combination of the input texture maps. Although both methods take an image-based approach, our method learns how appearance changes due to expression and view rather than prescribing it. Most importantly, these approaches are geared towards data-limited scenarios and, as a result, do not scale well in cases where data is more abundant. Our approach is designed specifically to leverage large quantities of high quality data to achieve the compelling depiction of human faces.

Because our method is concerned with view-dependent facial appearance, our work has a connection to others that have studied the view- and light-dependent appearance of materials and objects. One way to view this work is as a compression of the "8D reflectance function that encapsulates the complex light interactions inside an imaginary manifold via the boundary conditions on the surface of the manifold" [Klehm et al. 2015]. That is, we are capturing the lightfield at a manifold on which the cameras lie and compressing the appearance via a deconstruction of texture and geometry. Image-based rendering methods [Gortler et al. 1996; Kang et al. 2000] represent the light field non-parametrically and attempt to use a large number of samples to reconstruct novel views. The Bidirectional Texture Function [Dana et al. 1999] is a method to model both spatially-varying and view-varying appearance. These approaches have served as inspiration for this work.

To drive our model we develop a technique for learning a direct end-to-end mapping from images to deep appearance model codes. Although state of the art techniques for face registration also employ regression methods [Kazemi and Sullivan 2014; Xiong and la Torre Frade 2013], most approaches assume a complete view of the face where correspondences are easily defined through facial landmarks. An approach closest to our work is that of Olszewski et al. [2016], which employ direct regression from images to blendshape coefficients. The main difference to our work is in how corresponding expressions are found between the two domains. Whereas in Olszewski et al.'s work, correspondences are defined semantically through peak expressions and auditory aligned features of speech sequences, our approach does not require any human defined correspondence, or even that the data contains the same sequence of facial motions. Since we leverage unsupervised domain adaptation techniques, the only requirement of our method is that the distribution of expressions in both domains are comparable.

There have recently been a number of works that perform unsupervised domain adaptation using deep networks [Bousmalis et al. 2017; Kim et al. 2017; Liu et al. 2017; Zhu et al. 2017]. Many of these methods rely on Generative Adversarial Networks (GANs) to translate between two or more domains, sometimes with additional losses to ensure semantic consistency. Some works also use VAE in conjunction with other components to learn a common latent space between domains [Hsu et al. 2017; Liu et al. 2017]. Our approach differs from these works in that we leverage the pre-trained rendering VAE to help constrain the structure of the common latent space, resulting in semantics that are well matched between the modalities.

3 CAPTURING FACIAL DATA

To learn to render faces, we need to collect a large amount of facial data. For this purpose, we have constructed a large multi-camera capture apparatus with synchronized high-resolution video focused on the face. The device contains 40 machine vision cameras capable of synchronously capturing 5120×3840 images at 30 frames per second. All cameras lie on the frontal hemisphere of the face and are placed at a distance of about one meter from it. We use zoomed in

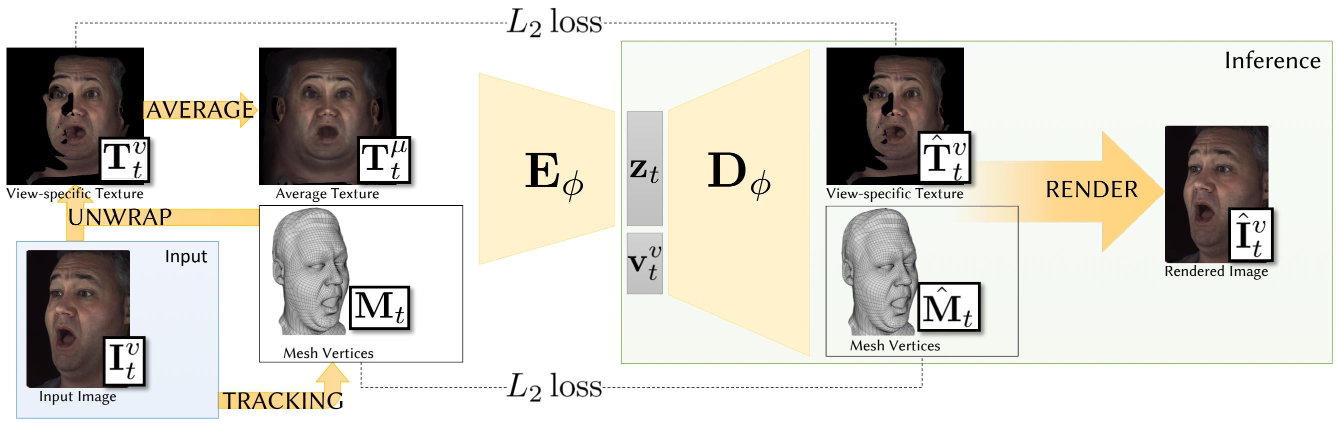

Fig. 2. Pipeline of our method. Our method begins with input images from the multi-camera capture setup. Given a tracked mesh, we can unwrap these images into view-specific texture maps. We average these texture maps over all cameras for each frame and input it and the mesh to a variational autoencoder. The autoencoder learns to reconstruct a mesh and view-specific texture because the network is conditioned on the output viewpoint. At inference time, we learn to encode a separate signal into the latent parameters **z** which can be decoded into mesh and texture and rendered to the screen.

50mm lenses, capturing pore-level detail, where each pixel observes about 50μm on the face.

We preprocess the raw video data by performing multiview stereo reconstruction. In order to achieve the best results, we evenly place 200 directional LED point lights directed at the face to promote uniform illumination.

To keep the distribution of facial expressions consistent across identities, we have each subject make a predefined set of 122 facial expressions. Each subject also recites a set of 50 phonetically-balanced sentences. The meta-data regarding the semantic content of the recordings is not utilized in this work, but its inclusion as additional constraints in our system is straightforward and is a potential direction for future work.

As input to our deep appearance model learning framework, we take the original captured images as well as a tracked facial mesh. To generate this data, we build personalized blendshape models of the face from the captured expression performances, similar to Laine et al. [2017], and use it to track the face through the captured speech performances by matching the images and the dense 3D reconstructions.

4 BUILDING A DATA-DRIVEN AVATAR

Our primary goal is to create extremely high-fidelity facial models that can be built automatically from a multi-camera capture setup and rendered and driven in real time in VR (90Hz). In achieving this goal, we avoid using hand-crafted models or priors, and instead rely on the rich data we acquired from our multiview capture apparatus.

We unify the concepts of 3D morphable models, image-based rendering, and variational autoencoders to create a real-time facial rendering model that can be learned from a multi-camera capture rig. The idea is to construct a variational autoencoder that jointly encodes geometry and appearance. In our model, the decoder outputs view-dependent textures—that is, a texture map that is "unwrapped" from a single camera image. It is view-specific and therefore contains view-dependent effects such as specularities, distortions due to imperfect geometry estimates, and missing data in areas of the face that are occluded.

The critical piece of the proposed method is that we use a conditional variational autoencoder to condition on the viewpoint of the viewer (at training time, the viewer is the camera from which the texture was unwrapped; at test time, the viewpoint we want to render from; in both cases the direction is composed with the inverse of the global head-pose in the scene), allowing us to output the correct view-specific texture. At test time, we can execute the decoder network in real-time to regress from latent encoding to geometry and texture and finally render using rasterization.

Figure 2 visualizes the training and inference pipeline of our algorithm. For this work, we assume that coarse facial geometry has been tracked and we use it as input to our algorithm with the original camera images. After geometry is tracked, we "unwrap" texture maps for each camera and for every frame of capture. Unwrapping is performed by tracing a ray from the camera to each texel of the texture map and copying the image pixel value into the texel if the ray is not occluded. These view-specific texture maps are what we want to generate at test time: reproducing them will cause the rendered image to match the ground truth image after rendering. To learn how to generate them efficiently, we use a conditional variational autoencoder (CVAE) [Kingma and Welling 2013]. Because we jointly model geometry and appearance, our autoencoder has two branches: an RGB texture map and a vector of mesh vertex positions.

Let $\mathbf{I}_t^v$ be an image from the multi-camera capture rig at time instant t from camera v (for a total of $\mathcal{V} = 40$ cameras). In this work, we assume that the viewpoint vector is relative to the rigid head orientation that is estimated from the tracking algorithm. At each time instant we also have a 3D mesh $\mathbf{M}_t$ (7306 vertices $\times 3 = 21918$-dimensional vector) with a consistent topology across time. Using

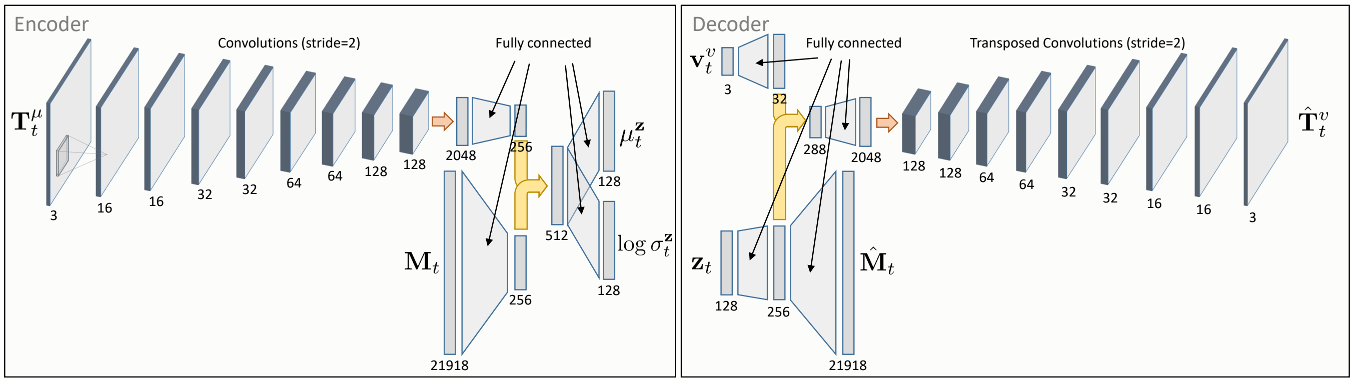

Fig. 3. Architecture of the Encoder and Decoder. The textures $\mathbf{T}_t^\mu$ and $\mathbf{T}_t^v$ are 3-channel 1024×1024 images. Each convolutional layer has stride 2 and the number of channels doubles after the first convolution every two layers. We combine the texture and mesh subencodings via concatenation. The decoder runs these steps in reverse: we split the network into two branches and use transposed convolutions of stride 2 to double the image resolution at every step. This decoder network executes in less than 5 milliseconds on an NVIDIA GeForce GTX 1080 GPU.

the image and mesh we can "unwrap" a view-specific texture map $\mathbf{T}_t^v$ by casting rays through each pixel of the geometry and assigning the intersected texture coordinate to the color of the image pixel. During training, the conditional VAE takes the tuple $(\mathbf{T}_t^\mu, \mathbf{M}_t)$ as input and $(\mathbf{T}_t^v, \mathbf{M}_t)$ as the target. Here,

$$\mathbf{T}_t^\mu = \frac{\sum_{v=1}^{\mathcal{V}} w_t^v \odot \mathbf{T}_t^v}{\sum_{v=1}^{\mathcal{V}} w_t^v}, \tag{1}$$

is the average texture, where w_t^v is a factor indicating whether each texel is occluded (0) or unoccluded (1) from camera v and $\odot$ represents an element-wise product. The primary reason for this asymmetry is to prevent the latent space from containing view information and to enforce a canonical latent state over all viewpoints for each time instant. The effects of this is discussed below.

The cornerstone of our method is the conditional variational autoencoder that learns to jointly compress and reconstruct the texture $\mathbf{T}_t^v$ and mesh vertices $\mathbf{M}_t$. This autoencoder can be viewed as a generalization of Principal Component Analysis (PCA) in conventional AAMs. The autoencoder consists of two halves: the encoder $\mathbf{E}_\phi$ and the decoder $\mathbf{D}_\phi$. The encoder takes as input the mesh vertices and texture intensities and outputs parameters of a Gaussian distribution of the latent space,

$$\mu_t^{\mathbf{z}}, \log \sigma_t^{\mathbf{z}} \leftarrow \mathbf{E}_\phi(\mathbf{T}_t^\mu, \mathbf{M}_t), \tag{2}$$

where the function $\mathbf{E}_\phi$ is parameterized using a deep neural network with parameters ϕ. At training time, we sample from the distribution,

$$\mathbf{z}_t \sim \mathcal{N}(\mu_t^{\mathbf{z}}, \sigma_t^{\mathbf{z}}), \tag{3}$$

and pass it into the decoder $\mathbf{D}_\phi$ and compute the reconstruction loss. This process approximates the expectation over the distribution defined by the encoder. The vector $\mathbf{z}_t$ is a data-driven low-dimensional representation of the subject's facial state. It encodes all aspects of the face, from eye gaze direction, to mouth pose, to tongue expression.

The decoder takes two inputs: the latent facial code $\mathbf{z}_t$ and the view vector, represented as the vector pointing from the center of the head to the camera $\mathbf{v}_t$ (relative to the head orientation that is estimated from the tracking algorithm). The decoder transforms the latent code and view vector into reconstructed mesh vertices and texture,

$$\hat{\mathbf{T}}_t^v, \hat{\mathbf{M}}_t \leftarrow \mathbf{D}_\phi(\mathbf{z}_t, \mathbf{v}_t^v), \tag{4}$$

where $\hat{\mathbf{T}}_t^v$ is the reconstructed texture and $\hat{\mathbf{M}}_t$ is the reconstructed mesh. After decoding, the texture, mesh, and camera pose information can be used to render the final reconstructed image $\hat{\mathbf{I}}_t^v$.

Figure 3 shows the architecture of the encoder and decoder. Conditioning is performed by concatenating the conditioning variable to the latent code $\mathbf{z}$ after each are transformed by a single fully connected layer. Since the mesh should be independent of viewpoint, it is only a function of the latent code. The texture decoder subnetwork consists of a series of strided transposed convolutions to increase the output resolution. In each layer, reparameterize the weight tensor using Weight Normalization [Salimans and Kingma 2016]. We use leaky ReLU activations between layers with 0.2 leakiness. The decoder network must execute in less than 11.1 milliseconds to achieve 90Hz rendering for real-time VR systems. We are able to achieve this using transposed strided convolutions even with a final texture size of 1024×1024. This is a major departure from most previous work for generating facial imagery that has been limited to significantly smaller output sizes.

Texture maps have non-stationary statistics that we can exploit in our network design. For example, DeepFace [Taigman et al. 2014] utilizes "locally-connected layers", a generalization of convolution where each pixel location has a unique convolution kernel. We utilize a similar but simpler approach: each convolutional layer has a bias that varies with both channel *and* the spatial dimensions. We find that this greatly improves reconstruction error and visual fidelity.

To train our system, we minimize the L_2-distance between the input texture and geometry and the reconstructed texture and geometry plus the KL-divergence between the prior distribution (an isotropic Gaussian) and the distribution of the latent space:

$$\ell(\phi) = \sum_{v,t} \lambda_T \left\| w_t^v \odot \left(\mathbf{T}_t^v - \hat{\mathbf{T}}_t^v \right) \right\|^2 + \lambda_M \left\| \mathbf{M}_t - \hat{\mathbf{M}}_t \right\|^2 + \lambda_Z \, \mathrm{KL}\Big(\mathcal{N}\left(\mu_t^{\mathbf{z}}, \sigma_t^{\mathbf{z}}\right) \,\big\|\, \mathcal{N}(0, \mathbf{I}) \Big), \tag{5}$$

where w_t^v is a weighting term to ensure the loss does not penalize missing data (i.e., areas of the face that are not seen by the camera) and λ_* are weights for each term (in all experiments we set these values to λ_T=1.0, λ_M=1.0, λ_Z=0.01). We use the Adam algorithm [Kingma and Welling 2013] to optimize this loss. Before training, we standardize the texture and geometry so that they have zero mean and unit variance. For each subject, our dataset consists of around 5,000 frames of capture under 40 cameras per frame. Training typically takes around one day for 200,000 training iterations with a mini-batch size of 16 on an NVIDIA Tesla M40.

During test time, we execute the decoder half to transform facial encodings $\mathbf{z}$ and viewpoint $\mathbf{v}$ into geometry and texture. Using the architecture shown in Figure 3, this takes approximately 5 milliseconds on an NVIDIA GeForce GTX 1080 graphics card; well within our target of 11.1 milliseconds. In practice, we find that in VR it is desirable to decode twice, creating one texture for each eye to induce parallax also in the generated texture. Our network is able to generalize in viewpoint enough that the small difference in viewpoint between the two eyes improves the experience by giving an impression of depth in regions that are poorly approximated by the sparse geometry (e.g., the oral cavity).

For our viewpoint conditioning to work, the decoder network $\mathbf{D}_\phi$ must rely on the viewpoint conditioning vector to supply all the information about the viewpoint. In other words, the latent code $\mathbf{z}$ should contain *no* information about the viewpoint of the input texture. This should be true for all types of variables that we may want to condition on at test time. The use of variational autoencoders [Kingma and Welling 2013] does promote these properties in the learned representation (as we will discuss in section 4.1). However, for the special case of viewpoint, we can explicitly enforce this factorization by supplying input $\mathbf{T}_t^\mu$ as the input texture, that is averaged across all cameras for a particular time instant t. This allows us to easily enforce a viewpoint-independent encoding of facial expression and gives us a canonical per-frame latent encoding.

4.1 Conditioned Autoencoding

A critical piece of our model is the ability to condition the output of the network on some property we want to control at test time. For our base model to work properly, we must condition on the viewpoint of the virtual camera. The idea of conditioning on some information we want to control at test time can be extended to include illumination, gaze direction, and even identity. We can broadly divide conditioning variables into two categories: extrinsic variables (e.g., viewpoint, illumination, etc.) and intrinsic variables (e.g., speech, identity, gaze, etc.).

4.1.1 Viewpoint Conditioning. The main variable that must be conditioned on is the virtual camera viewpoint. We condition on viewpoint so that at test time we will generate the appropriate texture from that viewer's point of view (relative to the avatars position and orientation). For this to work properly, the latent encoding $\mathbf{z}$ must not encode any view-specific information.

A viewpoint-independent expression representation can be achieved by using a variational autoencoder [Kingma and Welling 2013], which encourages the latent encoding $\mathbf{z}$ to come from a zero-mean unit-variance Gaussian distribution. It has been observed that variational autoencoders tend to learn a minimal encoding of the data because of the regularization of the latent space [Chen et al. 2016]. Because of this, the network is encouraged to use the conditioning variable (in this case, viewpoint) as much as possible in order to minimize the number of dimensions used in the encoding. Rather than rely only on this mechanism, we input a texture map averaged over all viewpoints to the encoder network to ensure a viewpoint-independent latent encoding $\mathbf{z}$. We will, however, exploit this mechanism for other conditioning variables.

4.1.2 Identity Conditioning. Avatars built with this system are person-specific. We would like to have a single encoder/decoder that models multiple people through an identity conditioning variable. With enough data, we may be able to learn to map single images to identity conditioning vectors, allowing us to dynamically create new avatars without needing a multi-camera capture rig.

Interestingly, with this identity conditioning scheme, we notice that the network learns an identity-independent representation of facial expression in $\mathbf{z}$ despite not imposing any correspondence of facial expression between different people. This is because the variational autoencoder (VAE) is encouraged to learn a common representation of all identities through its prior. Semantics tend to be preserved probably because the network can reuse convolutional features throughout the network.

5 DRIVING A DATA-DRIVEN AVATAR

Now that we are able to train the decoder to map latent encodings to texture and geometry, we can render animated faces in real-time. We would like the ability to drive our avatars with live or recorded performance rather than only playing back data from the original capture. In this section, we detail how to animate our avatar by controlling the latent facial code $\mathbf{z}$ with an unsupervised image-based tracking method. Because our focus is on creating the highest possible quality avatars and facial animation, we limit our scope to person-specific models (i.e., a person can drive only his or her own avatar).

The primary challenge for driving performance with these avatars is obtaining correspondence between frames in our multi-camera capture system and frames in our headset. Note that this is a unique problem for virtual reality avatars because one cannot wear the headset during the multi-camera capture to allow us to capture both sets of data simultaneously, as the headset would occlude the face. We address this problem by utilizing unsupervised domain adaptation.

5.1 Video-driven Animation

Our approach centers around image-based rendering on the rich multi-camera data to simulate headset images. The first step is to compute approximate intrinsic and extrinsic headset camera parameters in the multi-camera coordinate system (we do this by

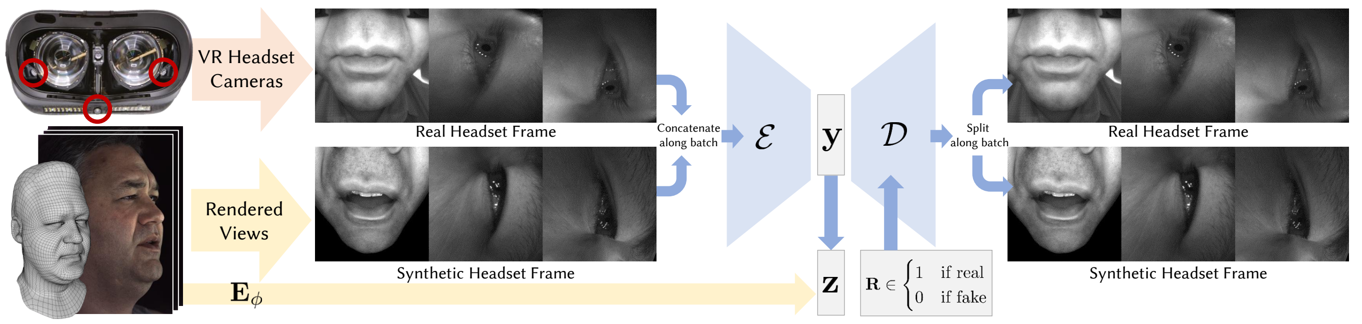

Fig. 4. Facial Tracking Pipeline. First, we generate synthetic headset images using image-based rendering on our multi-camera capture data. These images look geometrically like real headset images but not photometrically because of the difference in lighting. To account for this difference, we encode both synthetic headset images and real headset images using a VAE, which encourages learning a common representation of both sets. We can translate between the two modalities by flipping a conditioning variable.

hand for one frame and propagate the tracked head pose). Then, for each pixel of the synthetic headset image, we raycast into the tracked geometry and project that point into one of the multi-camera images to get a color value. This allows to produce synthetic images from the perspective of a headset with our multi-camera system.

Unfortunately, the lighting in the headset images and multi-camera images is quite different (and, in fact, of a different wavelength) so naively regressing from these synthetic headset images to facial encoding $\mathbf{z}$ will likely not generalize to real headset images.

There has been much work recently on performing unsupervised domain adaptation using adversarial networks that learn to translate images from one domain to another without any explicit correspondence (e.g., [Zhu et al. 2017]). One possibility for us is to use an image-to-image translation approach to transform synthetic headset images into real headset images and then learn a regression from translated synthetic headset images to our latent code $\mathbf{z}$. This scheme has two main drawbacks: first, the network learning the regression to latent code $\mathbf{z}$ never trains on real headset images; second, the adversarial component of these methods tend to be difficult to train.

We take the following alternative approach to solve this problem. First, we train a single variational autoencoder to encode/decode both real headset images and synthetic headset images. The Gaussian prior of the latent space will encourage the code $\mathbf{y}$ to form a common representation of both sets of images. We condition the decoder on a binary value indicating whether the image was from the set of real headset images or the set of synthetic images so that this information is not contained in the latent code. Next, we learn a linear transformation $\mathbf{A}_{\mathbf{y}\to\mathbf{z}}$ that maps the latent code $\mathbf{y}$ to the rendering code $\mathbf{z}$ for the synthetic headset images because we have correspondence between images of our multi-camera system $\mathbf{I}_t^v$ and latent codes $\mathbf{z}_t = \mathbf{E}_\phi(\mathbf{T}_t^\mu, \mathbf{M}_t)$ through our rendering encoder. If the VAE is successful in learning a common, semantically-consistent representation of real and synthetic headset images, then this linear regression will generalize to real headset images.

Note that while there is no guarantee that the semantics of the expression are the same when decoding in each of the two modalities, we observe that semantics tend to be preserved. We believe the primary reason for this is because the two image distributions are closely aligned and therefore the encoder network can make use of shared features.

Figure 4 shows a pipeline of our technique. In this pipeline, the encoder $\mathcal{E}$ takes one headset frame $\mathbf{H}_t$ consisting of three images, mouth $\mathbf{H}_t^m$, left eye $\mathbf{H}_t^l$, and right eye $\mathbf{H}_t^r$. Each headset frame is either real $\mathbf{H}_t^R$ or synthetic $\mathbf{H}_t^S$. The encoder $\mathcal{E}$ produces a latent Gaussian distribution,

$$\mu_t^{\mathbf{y}}, \log \sigma_t^{\mathbf{y}} \leftarrow \mathcal{E}(\mathbf{H}_t)\,. \tag{6}$$

At training time, we sample from this distribution to get a latent code,

$$\mathbf{y}_t \sim \mathcal{N}\left(\mu_t^{\mathbf{y}}, \sigma_t^{\mathbf{y}}\right), \tag{7}$$

as before. Our decoder $\mathcal{D}$ produces a headset frame given the latent code and an indicator variable :

$$\hat{\mathbf{H}}_t \leftarrow \mathcal{D}(\mathbf{y}_t, \mathbf{R}), \tag{8}$$

where $\mathbf{R} \in \{0, 1\}$ indicates whether the decoder should decode a real headset frame or a synthetic headset frame. This indicator variable allows the latent code to contain no modality-specific information because the decoder network can get this information from the indicator variable instead.

The architecture of our headset encoder $\mathcal{E}$ is as follows: for each of the three types of headset images (lower mouth, left eye, right eye), we create a three-branch network with each branch containing eight stride-2 convolutions with each convolution followed by a leaky ReLU with 0.2 leakiness. The number of output channels for the first layer is 64 and doubles every other convolutional layer. The three branches are then concatenated together and two fully-connected layers are used to output $\mu_t^{\mathbf{y}}$ and $\log \sigma_t^{\mathbf{y}}$. For our experiments, the latent vector $\mathbf{y}$ has 256 dimensions. The decoder network $\mathcal{D}$ is similar: three branches are created, with each branch consisting of a fully-connected layer followed by eight stride-2 transposed convolutions with each layer followed by a leaky ReLU with 0.2 leakiness. The first transposed convolution has 512 channels as input and this value halves every other transposed convolution. We condition the decoder by concatenting the conditioning variable $\mathbf{R}$ to every layer, replicating across all spatial dimensions for convolutional layers.

We don't use any normalization in these networks. Our encoder network runs in 10ms on an NVIDIA GeForce GTX 1080, which enables real-time tracking.

To train the network, we optimize the reconstruction loss, retargetting loss, and KL-divergence loss,

$$\ell(\theta) = \sum_t \lambda_H \left\|\mathbf{H}_t - \hat{\mathbf{H}}_t\right\|^2 + \lambda_A \left\|\mathbf{z}_t - \mathbf{A}_{\mathbf{y}\to\mathbf{z}}\mathbf{y}_t\right\|^2 + \lambda_Y \mathrm{KL}\Big(\mathcal{N}\left(\mu_t^{\mathbf{y}}, \sigma_t^{\mathbf{y}}\right) \,\big\|\, \mathcal{N}(0, \mathbf{I})\Big), \tag{9}$$

where $\mathbf{z}_t$ is known only for synthetic headset frames $\mathbf{H}^S$, $\mathbf{A}_{\mathbf{y}\to\mathbf{z}}$ linearly maps from tracking latent code $\mathbf{y}$ to rendering code $\mathbf{z}$, and $\lambda_\cdot$ are weights for each term of the loss (in all experiments we set λ_H=1.0, λ_A=0.1, and λ_Y=0.1). We optimize this loss using the Adam optimizer [Kingma and Ba 2014], as before. To make our method robust to the position of the HMD on the face, we perform a random translation, scaling, and rotation crop on the images. We also randomly scale the intensity values of the images to provide some robustness to lighting variation.

This completes the entire pipeline: a headset image is input to the headset encoding network $\mathcal{E}$ to produce the headset encoding $\mathbf{y}$, then it is translated to the multi-camera encoding $\mathbf{z}$, and finally it is decoded into avatar geometry $\hat{\mathbf{M}}$ and texture $\hat{\mathbf{T}}$ and rendered for the user. This architecture also works well for social interactions over a local- or wide-area network as only the latent code $\mathbf{z}$ needs to be sent across the network, reducing bandwidth requirements.

5.2 Artist Controls

In addition to driving the deep appearance model with video, we want to provide a system for rigging these models by character artists. To do this, we use a geometry-based formulation where we optimize to find a latent code that satisfies some geometric constraints, similar to Lewis and Anjyo [2010]. The artist can click vertices and drag them to a new location to create a constraint.

To find the latent code given the geometric constraints, we solve,

$$\arg\min_{\mathbf{z}} \sum_{i=1}^{N} \|\hat{\mathbf{M}}_i - \tilde{\mathbf{M}}_i\|^2 + \lambda_P \|\mathbf{z}\|^2, \tag{10}$$

where N is number of constraints, $\hat{\mathbf{M}} = \mathbf{D}_\phi(\mathbf{z})$ is the mesh resulting from decoding $\mathbf{z}$, $\hat{\mathbf{M}}_i$ is the vertex of constraint i, and $\tilde{\mathbf{M}}_i$ is the desired location for the vertex of constraint i. The regularization on $\mathbf{z}$ can be seen as a Gaussian prior, which is appropriate because the variational autoencoder encourages $\mathbf{z}$ to take on a unit Gaussian distribution (we set λ_P=0.01). Even though the constraint is only placed on geometry, the model will produce a plausible appearance for the constraints because geometry and appearance are jointly modeled.

6 RESULTS

In this section, we give quantitative and qualitative results of our method for rendering human faces and driving them from HMD cameras. First, we explore the effects of geometric coarseness, viewpoint sparsity, and architecture choice on avatar quality. Next, we demonstrate how our tracking variational autoencoder can preserve facial semantics between two modalities (synthetic and real) by switching the conditioning label. Finally, we show results of our complete pipeline: encoding HMD images and decoding and rendering realistic avatars.

6.1 Qualitative Results

Figure 5 shows results of our method's renderings compared to ground truth images for three different subjects. Our model is able to capture subtle details of motion and complex reflectance behavior by directly modeling the appearance of the face. Although some artifacts are present (e.g., blurring inside the mouth), they are less distracting than other types of facial rendering artifacts.

Figure 6 shows results of our multi-identity model. For this experiment, we used data from 8 different people and trained one network conditioned on a latent identity vector. Currently this model cannot generate plausible interpolations of people, but we believe this will be possible with significantly more data.

6.2 Effect of Geometric and Viewpoint Coarseness

A fundamental part of our model is that we jointly model geometry and texture. As a consequence, our model is able to "correct" for any bias in mesh tracking by altering the output of the texture maps to best match the true view-specific texture map. This powerful ability is part of what allows us to produce highly realistic avatars. There is a limit, however, to the ability of the texture network to resolve those errors. To explore this, we have constructed a series of experiments to analyze quantitatively and qualitatively how the model behaves when the geometry becomes coarser and viewpoints become sparser.

We investigate four different types of geometric coarseness: our original tracked geometry, significant smoothing of the original tracked geometry (we use 1000 iterations of Taubin smoothing for this [Taubin 1995]), a static ellipsoid subsuming the original mesh, and a billboard quad. We test billboard quad because image-based face generation in computer vision is typically performed on flat images. For each of these cases, we build our deep appearance model with different sets of training viewpoints (32 viewpoints, 16 viewpoints, 8 viewpoints, and 4 viewpoints) and compare performance in terms of mean-squared error on a set of 8 validation viewpoints held out from the total set of training viewpoints.

Figure 7 shows quantitative results for each of the four levels of geometric coarseness with four different sets of training viewpoints. The image-space mean-squared error (MSE), computed on the 8 validation viewpoints, shows a clear decrease in quality as the geometry is made more coarse. When the geometry input to our algorithm does not match the true geometry of the scene, the texture network must do more work to fill in the gaps, but there are limits to how well it generalizes. When the geometry fits well, the model performs well even with a small number of training viewpoints. When the geometry is too coarse, however, many viewpoints are needed for good generalization to new viewpoints.

Figure 8 shows qualitative results for the same experiment. This figure shows that accurate geometry (column "tracked") generalizes even when there are few views for training. Coarse geometry, (columns "ellipsoid" and "billboard"), however, generalizes poorly

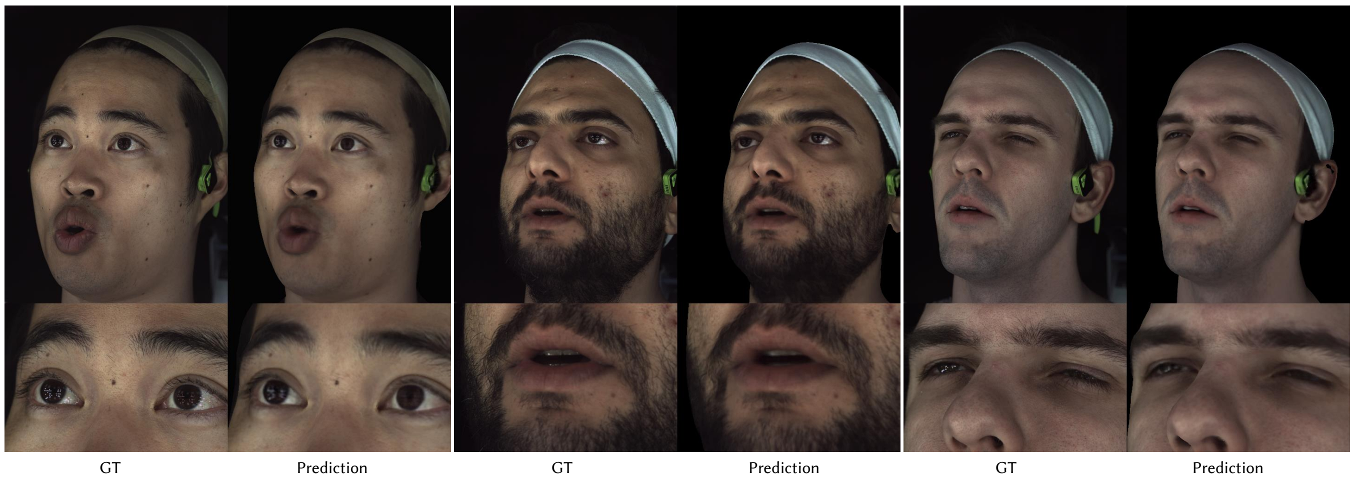

Fig. 5. Qualitative Results. This figure shows comparisons between the ground truth recordings and rendered mesh and texture predicted by our network (note that these frames were not in the training set) for four different subjects. We can see that our model achieves a high level of realism although it has some artifacts, in particular blurring within the mouth (we do not have tracked geometry inside the mouth, so our network must try to emulate that geometry with the texture). We find that these artifacts are less bothersome than traditional realtime human face renderings.

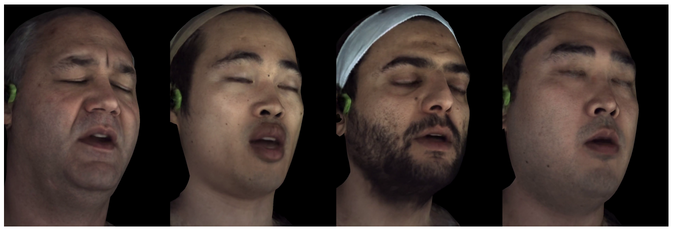

Fig. 6. Identity Conditioning. Here we demonstrate the automatic expressions correspondence obtained through identity conditioning. Here we decode the same latent code **z** into different people by conditioning on separate identity one-hot vectors.

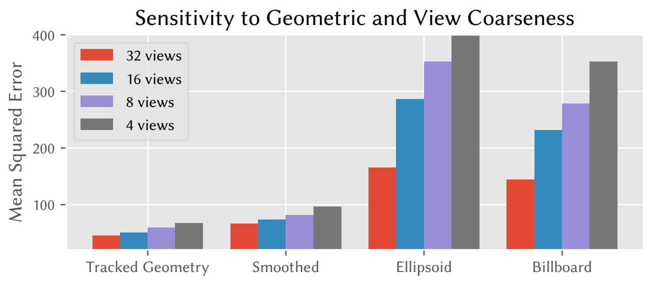

Fig. 7. Quantitative Effects of Geometric and Viewpoint Coarseness. We show the image-based mean-square error (MSE) for each geometry model and each set of training cameras (32, 16, 8, and 4 viewpoints). Here we see that accuracy of the geometric model allows the model to generalize well to the 8 validation viewpoints even when it has only seen 4 viewpoints at train time.

by introducing artifacts or by simply replicating the closest training viewpoint. Note that with coarse geometry the network has no problem learning the appearance of the training views but the generalization to new viewpoints is very poor.

6.3 Architecture Search

We perform an evaluation of different architectures for our render encoder and decoder networks. We primarily tested across two axes: conditioning techniques and normalization methods. For conditioning techniques we tested two categories of conditioning: conditioning on an early decoder layer by concatenation and conditioning on all decoder layers by addition. For normalization techniques, we tested no normalization, batch normalization [Ioffe and Szegedy 2015], and weight normalization [Salimans and Kingma 2016]. Finally, we investigate the advantage of a deep architecture over a linear or bilinear model. In each case, we report image-space mean-squared error for all valid head pixels. Here, head pixels are determined by raycasting into a dense stereo reconstruction of each frame.

6.3.1 Conditioning Methods. Figure 9 shows the results of our conditioning evaluation. The figure shows the reconstruction loss and the MSE for a series of experiments changing the conditioning method. We evaluate across two main axes: expanding the view vector to a larger dimensionality with a fully connected layer before conditioning, and conditioning on one early layer versus conditioning on all layers. Here, we found that conditioning on a single early layer in the decoder overall seems to outperform conditioning all layers in terms of the image-based mean-squared error on the set of 8 validation views. We also found that transforming the view vector before conditioning seems to be beneficial.

6.3.2 Normalization Methods. Figure 10 shows the results of our normalization evaluation. We tested three normalization methods: no normalization, batch normalization, and weight normalization.

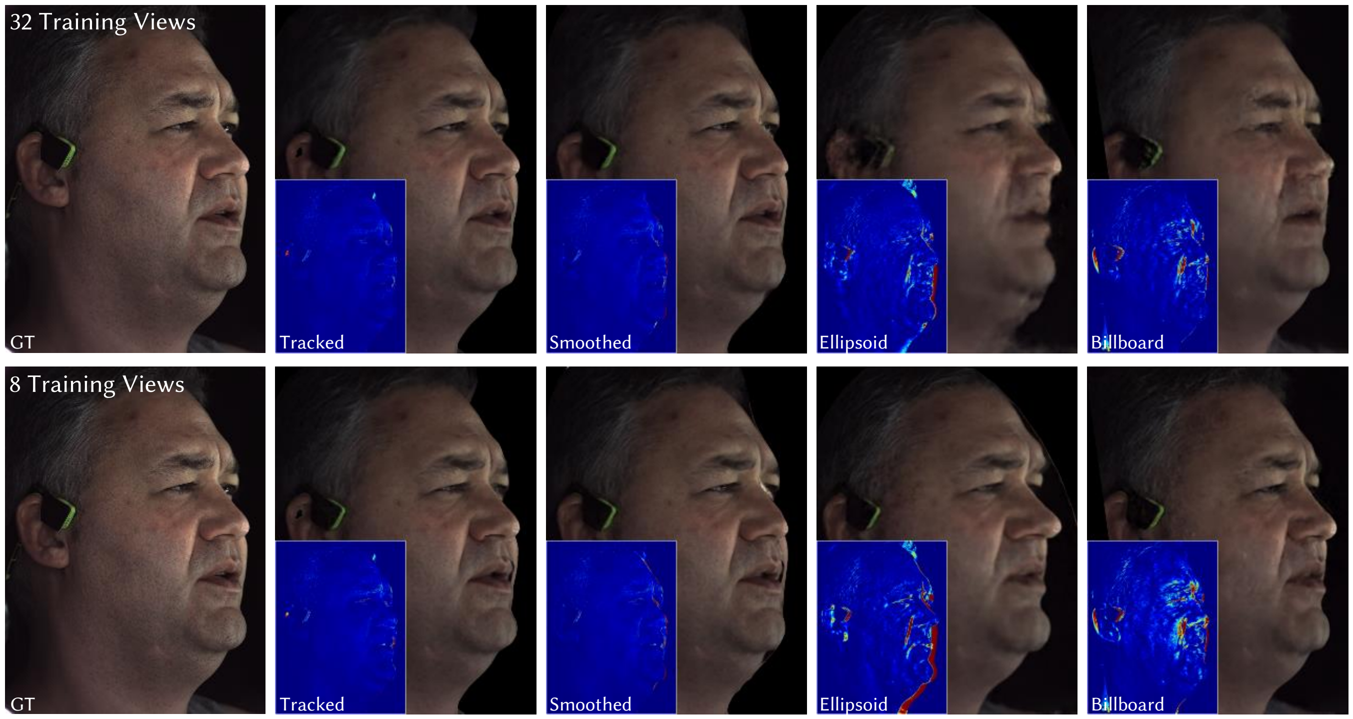

Fig. 8. Qualitative Results for Evaluating the Effects of Geometric and Viewpoint Coarseness. We show renderings of the avatar trained with four different geometry models and with two different sets of viewpoints. Each render has an inset showing the squared difference image from the ground truth. Note that the viewpoint shown was not seen during training. We see that when geometry matches the true geometry well, the rendering model generalizes to new viewpoints (see row "8 Training Views", column "Tracked"). When the geometry is too far from the true geometry, we start to see strange artifacts in the resulting image (see row "32 Training Views", column "Billboard").

In addition, we tested weight normalization without using per-channel-and-pixel biases (described in section 4). The figure shows that weight normalization far outperforms the other normalization techniques and that per-channel-and-pixel biases are very important for accuracy.

6.3.3 Linear AAM vs. Deep AAM. Aside from view conditioning, the main way we have augmented traditional AAMs is by changing the latent subspace from linear to nonlinear. The primary effect of this is to increase the expressiveness of the model although it also reduces the number of parameters. To evaluate the trade-off, we train our deep model compared to a linear model and a bilinear model (bilinear in view and facial state) with the same latent space dimensionality. Unfortunately a linear model requires $(21918 + 3 \cdot 1024^2) \cdot 128 \cdot 4 \cdot 2 = 3.24$GB of weights so we use only a 512×512 output resolution for the linear and bilinear models.

Table 1 shows quantitative results for linear and nonlinear representations. For each subject and method, we give the image-space mean-squared error. Our model has clear advantages over linear and bilinear models, including compactness in terms of number of parameters and reconstruction performance.

Table 1. Linear vs. Nonlinear representation. We evaluate our proposed method compared to a linear model (i.e., texture and geometry is a linear function of the latent code $\mathbf{z}$). Our model is not only better in terms of the image-based mean-squared error but also smaller in number of parameters (110MB vs. 806MB vs. 1.6GB).

	Model Type		
	Our method	Linear	Bilinear
Subject 1	**39.12**	59.00	69.17
Subject 2	**87.67**	121.44	128.94
Subject 3	**128.07**	173.33	168.16

6.4 Image-based vs. Texture/Geometry-based Loss

Traditional AAMs have been formulated as a PCA decomposition in terms of the texture and geometry jointly. By reformulating the AAM as an autoencoder we can actually make the loss a function of the reconstructed image $\hat{\mathbf{I}}_t^v$ itself. This is useful as it allows us to train on the metric on which we want to evaluate. It also opens the possibility of allowing the model to refine the output geometry over the input tracked geometry that may be erroneous.

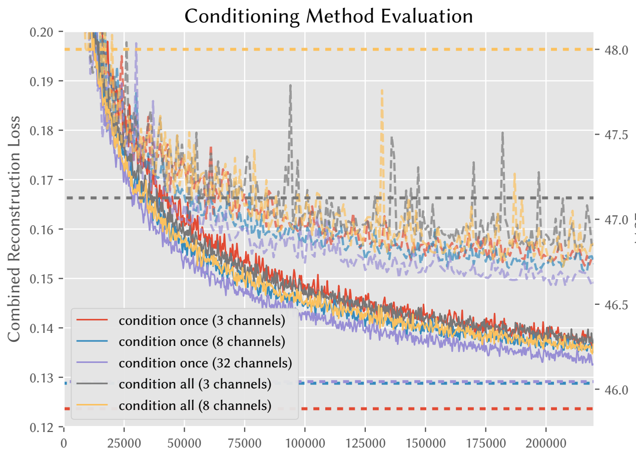

Fig. 9. Evaluation of Conditioning Method. We evaluate several ways to condition the network on viewpoint (red conditions the raw view vector on a single layer, blue conditions the view vector after a 3×8 fully-connected layer, purple conditions after a 3×32 fully-connected layer, grey conditions the raw view vector on all decoder layers, and yellow conditions the view vector after a 3×8 fully-connected layer on all decoder layers). Solid lines show training error (reconstruction loss), dashed lines show validation error (reconstruction loss), and flat dashed lines show image-space mean-squared error on a held-out set of cameras. Overall, conditioning on one early layer in the network tends to perform best.

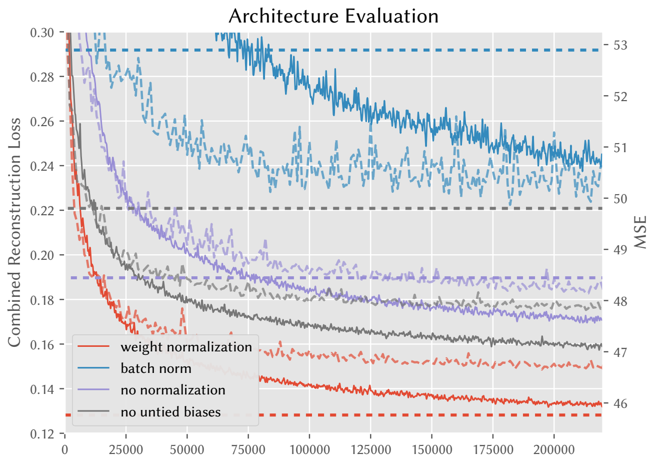

Fig. 10. Evaluation of Architectures. Solid lines show training error (reconstruction loss), dashed lines show validation error (reconstruction loss), and flat dashed lines show image-space mean-squared error on a held-out set of cameras. We found that using both weight normalization [Salimans and Kingma 2016] and per-channel-and-pixel biases yields the highest quality avatar.

To optimize with respect to the image-based reconstruction error, we rewrite the loss function as,

$$\ell(\phi) = \sum_{v,t} \lambda_I \left\| m_t^v \odot \left(\mathbf{I}_t^v - \hat{\mathbf{I}}_t^v\right)\right\|^2 + \lambda_T \left\| w_t^v \odot \left(\mathbf{T}_t^v - \hat{\mathbf{T}}_t^v\right)\right\|^2 + \lambda_M \left\| \mathbf{M}_t - \hat{\mathbf{M}}_t \right\|^2 + \lambda_Z \mathrm{KL}\Big(\mathcal{N}(\mu_t^z, \sigma_t^z) \Big\| \mathcal{N}(0, \mathbf{I})\Big), \tag{11}$$

Table 2. Image MSE for Different Training Loss Combinations. We train with two different training objectives (TG–texture + geometry loss, TGI–texture + geometry + image loss) to determine the optimal training objective. We found that using the texture + geometry + image loss produces the best results.

	Loss Type	
	TG	TGI
Subject 1	39.12	**33.66**
Subject 2	87.67	**75.48**

where $\hat{\mathbf{I}}_t^v$ is the reconstructed image from view v at time-instant t, and m_t^v is a mask that only penalizes pixels that contain the head (derived from the dense stereo reconstructions). We create a semi-differentiable rendering layer using two steps: the first step (non-differentiably) rasterizes triangle indices to an image; the second step (differentiably) computes texel coordinates for each pixel given the triangle indices, camera parameters, and mesh vertices and samples the texture map at the coordinate computed for each pixel. This layer allows us to backpropagate through the parameters of our model and train the system end-to-end.

To evaluate how an image-based loss during training may improve the results we compare two different models: a model optimized with texture+geometry loss and a model optimized with texture+geometry+image loss. We found that without a geometry term, the loss becomes unstable and the geometry tends to lose its structure.

Table 2 shows the quantitative results from these experiments. For each combination of training loss (TG–texture + geometry, TGI–texture + geometry + image) we give the image-based loss on set of 8 validation viewpoints. Incorporating the image-based loss certainly helps lower the MSE although qualitatively the results are similar. We also found that training was unstable without the geometry loss. This is likely because there is not enough long-range signal in the image gradients to effectively estimate geometry.

6.5 Video-driven Animation

In this section, we show how our tracking VAE ($\mathcal{E}, \mathcal{D}$) builds a common representation of facial state across two different modalities and then we give qualitative results showing our full live animation pipeline. Note that in this work our pipeline is entirely person-specific.

Figure 11 shows examples of encoding with one modality and decoding with the other. We can see in these examples that semantics tend to be preserved when the modality is changed at inference time despite only encoding and decoding the same modality at train time. This is primarily due to the variational autoencoder prior, which encourages the latent space to take on a unit Gaussian distribution.

Figure 12 shows qualitative results of a user driving an avatar in real time. Our system works well not only for expressions but also for subtle motion during speech. The benefit of our model is that it encodes facial state with high accuracy, encoding it into a joint geometry and appearance model. This allows us to model complex

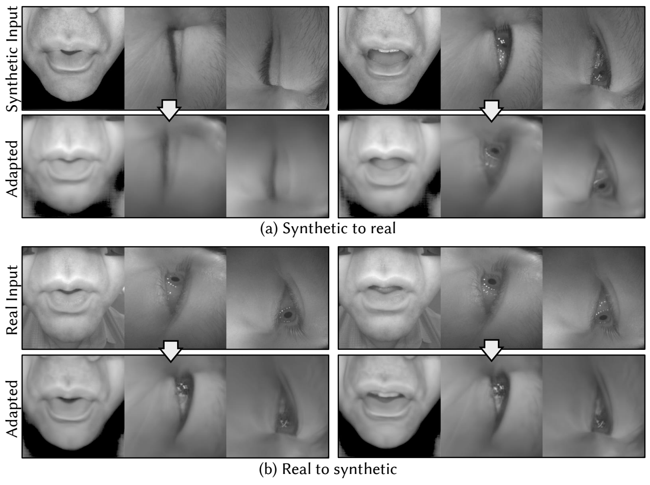

Fig. 11. Translating between real headset images to synthetic headset images. Sub-figure (a) shows synthetic headset images (top row) translated to real headset images (bottom row) by switch the conditioning variable in the VAE. Sub-figure (b) shows real headset images (top row) translated to synthetic headset images (bottom row) with the same method). This shows that the y contains a common representation of facial state regardless of being synthetic or real.

changes in appearance due to changes in blood flow, complex materials, and incorrect geometry estimates. Please see our supplemental video for additional examples.

7 DISCUSSION

In this paper, we presented a method for capturing and encoding human facial appearance and rendering it in real-time. The method unifies the concepts of Active Appearance Models, view-dependent rendering, and deep networks. We showed that the model enables photo-realistic rendering by leveraging multiview image data and that we can drive the model with performance. We believe there are exciting opportunities for extending our approach in different ways.

Our approach is unique because we directly predict a shaded appearance texture with a learned function. This is in contrast to traditional approaches that predict physically-inspired lighting model parameters (e.g., albedo maps, specular maps) which enable relighting. A limitation of our approach in its current form is a limited ability to relight. With improvements to our capture apparatus to allow for dynamic lighting during capture, we believe we can alleviate these limitations by introducing lighting as a conditioning variable.

As our experiments showed, we are able to accurately predict the appearance of the face when our tracked geometry closely matches the true surface of the face. This approach is even effective for regions of the face that are typically very difficult to accurately model because of their complex reflectance properties (e.g., eyes and teeth). Some artifacts still remain, however (e.g., blurring on the teeth) especially where there is no true smooth surface (e.g., hair). We may be able to turn to alternative representations of geometry (e.g., point clouds) to alleviate some of these problems.

Our tracking model enables high-fidelity real-time tracking from cameras mounted on a virtual reality headset by automatic unsupervised correspondence between headset images and multi-camera capture images. In this work, we limited the scope to building person-specific trackers and rendering models. To increase the scalability of our work, we need to modify our approach to support tracking and rendering for arbitrary people. This is a difficult problem because it requires learning semantic facial correspondence between different people.

Finally, we believe that building realistic avatars of the entire body with our approach will help enable self- and social- presence in virtual reality. This task comes with a new set of problems: handling the dynamics of clothing, the difficulties of articulating limbs, and modeling the appearance of interactions between individuals. We are confident that these are tractable problems along this line of research.

REFERENCES

Volker Blanz and Thomas Vetter. 1999. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 187–194. https://doi.org/10.1145/311535.311556

S. M. Boker, J. F. Cohn, B. J. Theobald, I. Matthews, M. Mangini, J. R. Spies, Z Ambadar, and T. R. Brick. 2011. Motion Dynamics, Not Perceived Sex, Influence Head Movements in Conversation. *J. Exp. Psychol. Hum. Percept. Perform.* 37 (2011), 874–891.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. (07 2017), 95–104.

Chen Cao, Hongzhi Wu, Yanlin Weng, Tianjia Shao, and Kun Zhou. 2016. Real-time Facial Animation with Image-based Dynamic Avatars. *ACM Trans. Graph.* 35, 4, Article 126 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925873

Dan Casas, Andrew Feng, Oleg Alexander, Graham Fyffe, Paul Debevec, Ryosuke Ichikari, Hao Li, Kyle Olszewski, Evan Suma, and Ari Shapiro. 2016. Rapid Photorealistic Blendshape Modeling from RGB-D Sensors. In *Proceedings of the 29th International Conference on Computer Animation and Social Agents (CASA '16)*. ACM, New York, NY, USA, 121–129. https://doi.org/10.1145/2915926.2915936

S. A. Cassidy, B. Stenger, K. Yanagisawa, R. Cipolla, R. Anderson, V. Wan, S. Baron-Cohen, and L Van Dongen. 2016. Expressive Visual Text-to-Speech as an Assistive Technology for Individuals with Autism Spectrum Conditions. *Computer Vision and Image Understanding* 148 (2016), 193–200.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational Lossy Autoencoder. *CoRR* abs/1611.02731 (2016). arXiv:1611.02731 http://arxiv.org/abs/1611.02731

Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. 2001. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 6 (June 2001), 681–685.

Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. 1999. Reflectance and Texture of Real-world Surfaces. *ACM Transactions on Graphics* 18, 1 (Jan. 1999), 1–34.

G. J. Edwards, C. J. Taylor, and T. F. Cootes. 1998. Interpreting Face Images Using Active Appearance Models. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition (FG '98)*. IEEE Computer Society, Washington, DC, USA, 300–.

P. Ekman. 1980. *The Face of Man: Expressions of Universal Emotions in a New Guinea Village.* Garland Publishing, Incorporated.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1996)*. ACM, New York, NY, USA, 43–54.

X. Hou, L. Shen, K. Sun, and G. Qiu. 2017. Deep Feature Consistent Variational Autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 1133–1141.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2017. Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data. In *NIPS*. 1876–1887.

Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. 2017. Avatar Digitization from a Single Image for Real-time Rendering. *ACM Trans. Graph.* 36, 6, Article 195 (Nov. 2017), 14 pages. https://doi.org/10.1145/3130800.31310887

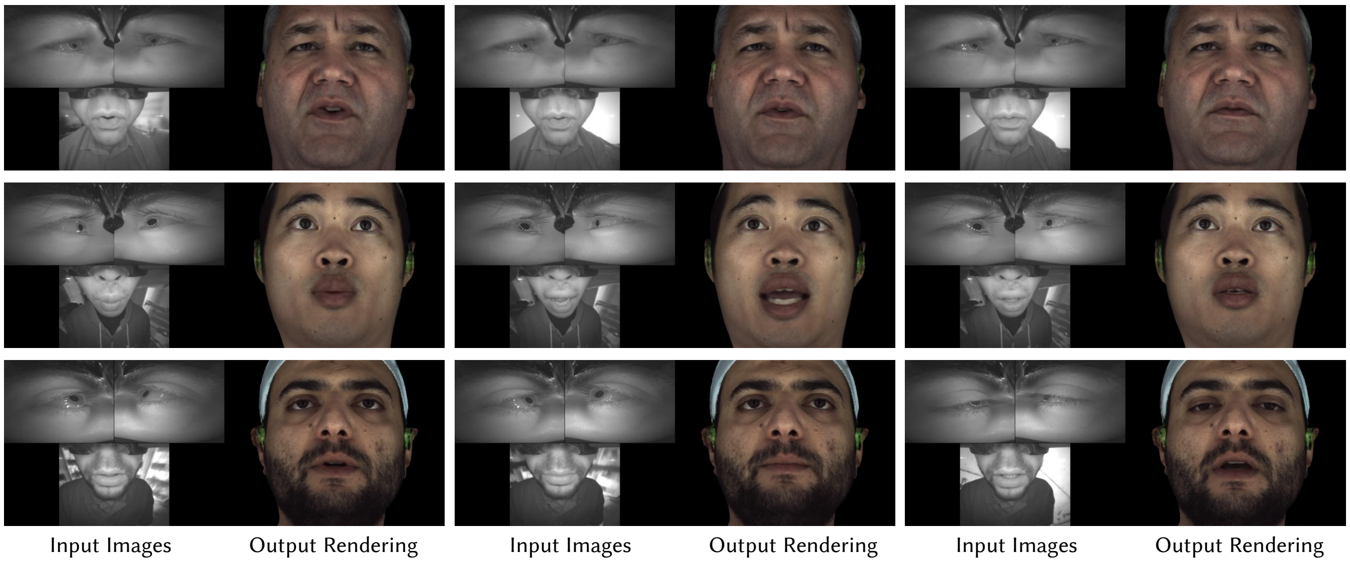

Fig. 12. Video-driven Animation. This figure shows three examples per subject of HMD images (left) encoded into the latent code **z** and decoded into geometry and appearance and rendered (right). Despite only seeing parts of the face, our animation method captures and reproduces facial state well.

Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. 2015. Dynamic 3D Avatar Creation from Hand-held Video Input. *ACM Transactions on Graphics* 34, 4, Article 45 (July 2015), 14 pages.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.. In *ICML (JMLR Workshop and Conference Proceedings)*, Francis R. Bach and David M. Blei (Eds.), Vol. 37. JMLR.org, 448–456. http://dblp.uni-trier.de/db/conf/icml/icml2015.html#IoffeS15

Sing Bing Kang, R. Szeliski, and P. Anandan. 2000. The geometry-image representation tradeoff for rendering. In *Proceedings 2000 International Conference on Image Processing*, Vol. 2. 13–16 vol.2.

Vahid Kazemi and Josephine Sullivan. 2014. One Millisecond Face Alignment with an Ensemble of Regression Trees. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *ICML*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations* abs/1412.6980 (2014).

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations.*

Oliver Klehm, Fabrice Rousselle, Marios Papas, Derek Bradley, Christophe Hery, Bernd Bickel, Wojciech Jarosz, and Thabo Beeler. 2015. Recent Advances in Facial Appearance Capture. *Computer Graphics Forum (Proceedings of Eurographics)* 34, 2 (May 2015), 709–733.

Reinhard Knothe, Brian Amberg, Sami Romdhani, Volker Blanz, and Thomas Vetter. 2011. *Morphable Models of Faces.* Springer London, London, 137–168.

Tejas D. Kulkarni, William F. Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. 2015. Deep Convolutional Inverse Graphics Network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15).* MIT Press, Cambridge, MA, USA, 2539–2547.

Samuli Laine, Tero Karras, Timo Aila, Antti Herva, Shunsuke Saito, Ronald Yu, Hao Li, and Jaakko Lehtinen. 2017. Production-level Facial Performance Capture Using Deep Convolutional Neural Networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '17).* ACM, New York, NY, USA, Article 10, 10 pages.

J.P. Lewis and Ken Anjyo. 2010. Direct Manipulation Blendshapes. *IEEE Computer Graphics and Applications* 30, 4 (2010), 42–50.

John P. Lewis, Ken ichi Anjyo, Taehyun Rhee, Mengjie Zhang, Frédéric H. Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. In *Proc. Eurographics State of The Art Report.*

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised Image-to-Image Translation Networks. In *NIPS.*

Iain Matthews and Simon Baker. 2004. Active Appearance Models Revisited. *International Journal of Computer Vision* 60, 2 (Nov. 2004), 135–164.

Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. 2016. High-Fidelity Facial and Speech Animation for VR HMDs. *Proceedings of ACM SIGGRAPH Asia 2016* 35, 6 (December 2016).

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015). http://arxiv.org/abs/1511.06434

Tim Salimans and Diederik P Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 901–909.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14).* IEEE Computer Society, Washington, DC, USA, 1701–1708.

G. Taubin. 1995. Curve and Surface Smoothing Without Shrinkage. In *Proceedings of the Fifth International Conference on Computer Vision (ICCV '95).* IEEE Computer Society, Washington, DC, USA, 852–.

J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2018. FaceVR: Real-Time Gaze-Aware Facial Reenactment in Virtual Reality. *ACM Transactions on Graphics 2018 (TOG)* (2018).

Georgios Tzimiropoulos, Joan Alabort-i Medina, Stefanos Zafeiriou, and Maja Pantic. 2013. *Generic Active Appearance Models Revisited.* Springer Berlin Heidelberg, Berlin, Heidelberg, 650–663.

Xuehan Xiong and Fernando De la Torre Frade. 2013. Supervised Descent Method and its Applications to Face Alignment. In *IEEE International Conference on Computer Vision and Pattern Recognition.* Pittsburgh, PA.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. (December 2017).