The Eyes Have It: An Integrated Eye and Face Model for Photorealistic Facial Animation

GABRIEL SCHWARTZ, Facebook Reality Labs
SHIH-EN WEI, Facebook Reality Labs
TE-LI WANG, Facebook Reality Labs
STEPHEN LOMBARDI, Facebook Reality Labs
TOMAS SIMON, Facebook Reality Labs
JASON SARAGIH, Facebook Reality Labs
YASER SHEIKH, Facebook Reality Labs

Fig. 1. **Avatars rendered and driven by our system.** These images highlight the quality of the renders produced by our system. On the left, we show renders of the learned avatars, and on the right, a view of the avatar as driven in our real-time system.

Interacting with people across large distances is important for remote work, interpersonal relationships, and entertainment. While such face-to-face interactions can be achieved using 2D video conferencing or, more recently, virtual reality (VR), telepresence systems currently distort the communication of eye contact and social gaze signals. Although methods have been proposed to redirect gaze in 2D teleconferencing situations to enable eye contact, 2D video conferencing lacks the 3D immersion of real life. To address these problems, we develop a system for face-to-face interaction in VR that focuses on reproducing photorealistic gaze and eye contact. To do this, we create a 3D virtual avatar model that can be animated by cameras mounted on a VR headset to accurately track and reproduce human gaze in VR. Our primary contributions in this work are a jointly-learnable 3D face and eyeball model that better represents gaze direction and upper facial expressions, a method for disentangling the gaze of the left and right eyes from each other and the rest of the face allowing the model to represent entirely unseen combinations of gaze and expression, and a gaze-aware model for precise animation from headset-mounted cameras. Our quantitative experiments show that our method results in higher reconstruction quality, and qualitative results show our method gives a greatly improved sense of presence for VR avatars.

CCS Concepts: • **Computing methodologies → Neural networks**; **Virtual reality**.

Additional Key Words and Phrases: Differentiable Rendering, Eye Modeling

ACM Reference Format:
Gabriel Schwartz, Shih-En Wei, Te-Li Wang, Stephen Lombardi, Tomas Simon, Jason Saragih, and Yaser Sheikh. 2019. The Eyes Have It: An Integrated Eye and Face Model for Photorealistic Facial Animation. *ACM Trans. Graph.* 38, 6, Article 91 (November 2019), 15 pages. https://doi.org/10.1145/3386569.3392493

Authors' addresses: Gabriel Schwartz, Facebook Reality Labs, gbschwartz@fb.com; Shih-En Wei, Facebook Reality Labs, swei@fb.com; Te-Li Wang, Facebook Reality Labs, teli@fb.com; Stephen Lombardi, Facebook Reality Labs, stephen.lombardi@fb.com; Tomas Simon, Facebook Reality Labs, tomas.simon@fb.com; Jason Saragih, Facebook Reality Labs, jason.saragih@fb.com; Yaser Sheikh, Facebook Reality Labs, yasers@fb.com.

1 INTRODUCTION

Eye contact is a strong and important social signal [Chen 2002], and humans can accurately estimate where someone's eyes are pointing just by looking at them [Cline 1967; Gibson and Pick 1963]. Recent efforts in image-space gaze-correction [Kononenko et al. 2018; Wolf et al. 2010] demonstrate the importance of achieving eye-contact in a telepresence application, but such methods are motivated by the

inherent conflict between looking at the screen (your conversation partner) and the camera (for perceived eye-contact). A photorealistic avatar viewed in 3D, e.g., in virtual reality, offers the unique benefit that eye contact can potentially be enabled by simply looking at someone's eyes.

Our goal is to build a system that enables remote communication for anyone across the planet. Unfortunately, existing avatar technology is limited in these areas. Conventional computer graphics rigs and rendering approaches can produce an avatar of high quality, but require significant manual artist work and are therefore costly to scale to the general population. Image-space generative models, such as StyleGAN [Karras et al. 2018], can synthesize extremely crisp photorealistic images of faces, and do so at large scales in an automated fashion. Reliably controlling the output of these unsupervised methods, however, is still an open research problem.

Recently, Lombardi et al. [2018] proposed a hybrid of machine learning and conventional graphics methods to render and control photorealistic avatars in real-time and Wei et al. [2019] introduced an improved method for driving these avatars. There are, however, numerous problems with the quality of the appearance of the eyes in these models: novel combinations of gaze and expression produce an uncanny appearance of the face, the fixation of the eyes cannot be explicitly controlled, and where it can be controlled it cannot verge to novel depths. All of these factors lead to a poor gaze representation and eye appearance, making eye contact impossible. We will propose and evaluate a number of potential improvements to this model's generalization behavior, but on their own, each improvement in generalization comes at the cost of poorer reconstruction quality.

To achieve the best of both worlds, we propose a novel model for joint eye and face appearance that leverages the strengths of machine learning while drawing inspiration from conventional computer graphics models of the eye to improve generalization *and* reconstruction quality. Our model combines an improved geometric representation of the eye with the losses described above to ensure that the surrounding facial mesh matches the eye's behavior. In order to ensure a seamless transition between this new eyeball geometry and the rest of the face, we jointly optimize the full facial appearance using differentiable rendering to match observed images. The result is a model with precise and direct control over apparent gaze, and improved fidelity of eye appearance.

Not only can we render eyes and faces with higher quality, we can also drive them in real-time using VR headset-mounted cameras (HMC). As Wei et al. [2019] showed, obtaining high-quality correspondences between the capture stage and real-time driving domains is important if we are to drive our avatar. One of our contributions in this work is to provide a novel avenue for cross-domain correspondence, in the form of face state and explicit gaze directions. We show that if we have gaze information in the capture stage domain, and likewise in the real-time domain, we may use these corresponding gaze directions to ensure that the driven avatar respects the driver's eye orientation far more precisely than previous methods. This is one condition required to provide a sense of eye contact.

2 RELATED WORK

2.1 Modeling Human Face and Eyes

There is a long history in computer graphics of creating high-fidelity rigs of human faces [Alexander et al. 2009; Bergeron and Lachapelle 1985; Porter 1997] although few approaches focus primarily on the eyes [Francois et al. 2009]. Recently, however, Bérard et al. [2014] developed a high-quality capture system and model for human eyes. This work was the first major attempt to model all parts of the eye from an image-based capture system. Bérard et al. [2016] later extended this work by enabling fitting of the eye model from a less-constrained capture setup. The approach in this work is similar to ours as both methods use multi-view data to estimate an eye model. Rather than fitting just to features extracted from the images, however, our method learns a face and eye model by matching image data pixel-for-pixel using differentiable rendering to better achieve realism [Kato et al. 2018; Liu et al. 2019; Loper and Black 2014]. Wen et al. [2017]; Wood et al. [2016] do optimize their gaze-focused models using pixel losses, but do not aim for both photorealism and realtime applications.

Many recent approaches in machine learning attempt to model the face and eyes in image-space [Karras et al. 2018], some with explicit controls on the output images [Chen et al. 2016; Radford et al. 2016]. Although these approaches generate extremely realistic images, they do not model the 3D geometry of the face and therefore it is difficult to explicitly produce smooth, realistic changes in viewpoint.

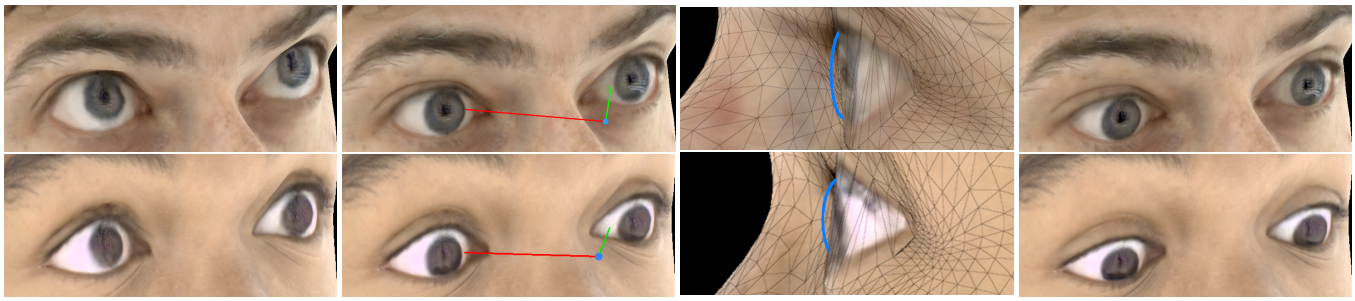
Some methods incorporate additional 3D geometry to alleviate this. Cao et al. [2016] produce an image-based avatar with a morphable head and hair model with billboards to represent the eyes and inner mouth. Though they explicitly model eyes, they are not full 3D models, nor are they learned from images, which limits the realism of their appearance.

Lombardi et al. [2018] proposed a data-driven model of the face learned from multi-view image data with tracked mesh. The main idea is to use a variational autoencoder [Kingma and Welling 2013] to jointly model geometry and view-dependent texture, similar to active appearance models [Cootes et al. 1998]. A main feature of the work is that it is heavily data-driven, and therefore can reproduce realistic facial appearance and motion. We improve upon this model by addressing many of its shortcomings with respect to the eyes. While doing so, we develop a simple gaze tracking approach to enable explicit control of the eyeball model. Compared to approaches that attempt to track gaze in the wild using a limited set of cameras [Fischer et al. 2018; Park et al. 2019; Ranjan et al. 2018], our approach is more specialized and simplified because the environment is controlled.

2.2 Face and Eye Tracking for VR Animation

Tracking human faces and eyes in VR is challenging as they are largely occluded by the headset itself. This makes it difficult to find correspondences between images captured from the headset renders of the avatar.

Olszewski et al. [2016] regress from mouth and eye images captured by headset-mounted cameras to animation parameters whose correspondence is learned by performing dynamic time-warping between headset training data and non-headset training data. This



(a) Expression with Unseen Gaze (b) Unseen Vergence Depth (c) Missing Geometry (d) Rare Expression

Fig. 2. **Failure Cases of Deep Appearance Models.** (a) Rendering expressions with novel gaze directions not observed during training may produce uncanny distortions and blurring. (b) The model may fail to converge the eyes at distances other than those observed during training (lines represent the gaze vectors given as input). (c) As their model lacks geometry on the eyeball surface, it fails to generalize to novel views (missing geometry in blue). (d) For rare expressions, the eye fidelity is lower than for more common expressions, particularly in the sharpness of glints (or lack thereof). Any one of these failures can result in a model which fails to evoke a feeling of eye contact.

method produces non-photorealistic avatars. FaceVR [Thies et al. 2016] builds a personalized avatar model from images, and a camera outside the headset is used to track a blendshape model of the face. The eyes are represented using an image retrieval approach, whereby an appropriate exemplar is found in the training database based on expression and gaze direction.

Wei et al. [2019] presented a method to find correspondences between HMC images and the latent state of photoreal avatars based on the idea of analysis-by-synthesis and multiview image style transfer. By minimizing a self-supervised reconstruction loss in the image domain through differentiable rendering, avatar animation quality and realism is greatly improved. Their use of generative adversarial networks (GANs), however, requires complicated distribution matching schemes, and slight mismatches between distributions can cause gaze directions in the image to be altered during style transfer. This is hard to fix because the avatar does not allow any explicit eye control.

In this work, we improve this method by performing style transfer in texture space instead of image space, and we estimate explicit gaze directions from HMC images for driving the proposed eyeball model. While this component is similar to many eye tracking systems in VR, such as Tobii VR [2018], in this work we present a model that jointly predicts personalized gaze and facial expression in realtime.

3 THE PATH TO EYE CONTACT

Our goal is to build a system to enable virtual telepresence, using photorealistic avatars, at scale, with a level of fidelity sufficient to achieve eye-contact. Physically inspired representations [Seymour et al. 2017], can generate renderings with high realism, but heavily depend on accurate estimates of geometry and reflectance properties of the eye and periocular region that are challenging to acquire automatically in practice.

Recent data-driven representations [Lombardi et al. 2018] obviate the need for highly accurate geometry by simulating view-dependent effects on imperfect geometry using neural networks. As shown in Fig. 2, however, these approaches generalize poorly to viewpoints, gaze directions and gaze-expression combinations not seen during training. As a result, this approach fails to capture real gaze and eye contact necessary for achieving a sense of social presence during during telepresence.

Despite its aforementioned shortcomings, the method proposed by Lombardi et al. [2018] can offer simple scalability if we can address its shortcomings regarding generalization. Their method accurately reproduces conditions seen during training, but cannot do the same for unseen or rare conditions, such as novel vergence combinations. This range of eye motion, while rare in their capture process, is common in natural situations. A failure to reproduce it can lead to uncanny interactions.

We address this limitation in a two-fold approach. First, we separate the facial model into spatially disjoint regions such that the eyes can be controlled independently. To increase the precision with which we may control the eyes, we replace the wholly-uninterpretable latent space with one where gaze information is treated as a separate signal which we may provide directly. This gives us more precise gaze control, and also introduces an additional avenue for correspondence between the capture system and realtime headset domains.

These improvements, applied on their own, allow the avatar representation to better represent novel vergence and gaze/expression combinations at the cost of overall reconstruction quality. To counteract this, we employ an explicit eyeball model (EEM) that better captures eye geometry and motion. Simply applying existing methods to learn this new representation results in artifacts where the eye meets the coarse face mesh. We optimize our model using differentiable inverse rendering to allow the mesh and texture to best-explain observed images beyond the available geometric supervision. The full process is described in Fig. 3 and in more detail in §4 and §5.

To drive this joint eye and face model in realtime, we build on the method of Wei et al. [2019]. We take advantage of our newly-introduced gaze signals as an additional avenue for correspondence between headset images and rendered avatars, allowing us to measure how well the avatar's gaze is matching the user's appearance.

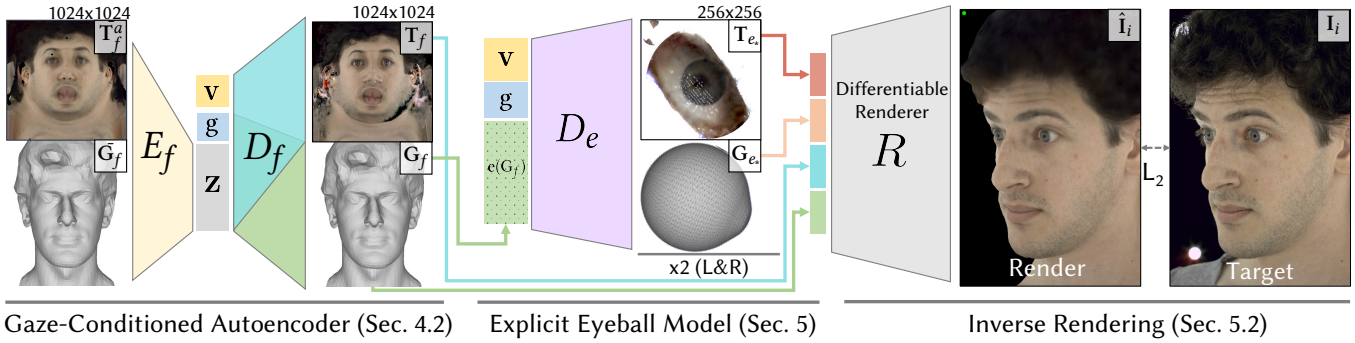


Fig. 3. **Joint Eye and Face Appearance Model.** Given an encoding of gaze-independent facial expression $\mathbf{z}$ (§4.2, §5.2.1), along with estimated left/right gaze vectors $\mathbf{g}$ (§4.1), and viewpoint $\mathbf{v}$, we produce geometry and view-dependent texture for the face with independent left/right eye control (§4.3), as well as a separate texture for the eyeball (§5.1). We orient a simple geometric model for the eyeball (§4.1) based on gaze estimates, and render the entire eye and face model differentiably. We optimize the system with inverse rendering to match captured images. In §6, we show that we may drive this model in real-time from a VR headset.

We simplify their approach, which relied image-space style translation, with a texture-space conversion from color avatar textures to IR headset textures. Thanks to our improvements, the avatar's gaze can be controlled independently and, by predicting independent gaze signals from left and right eye cameras in the headset, we can ensure the model generalizes well to novel fixation directions.

4 AVATAR GAZE CONTROL

We start by proposing a series of modifications to deep appearance models allowing them to reconstruct new vergence conditions that are unseen in the training data but are crucial for real social interactions. Each of the proposed steps will improve generalization at the cost of fidelity, but in §5, we introduce a model that more than counteracts this degradation. To support these investigations, we will first augment an existing training dataset with gaze tracking.

4.1 Multi-View Gaze Estimation

In this work, we would like to provide gaze as a conditioning variable for the decoder to enable explicit gaze control and also to use it for cross-domain correspondence in later sections. To do so, we must first estimate it from multi-view training images by tracking the subject's eyes. For each frame, we detect 8 keypoints along the limbus in 12 different input views using a detector similar to Li et al. [2019]. We then fit a geometrical eye model to each sequence, including a per-frame estimate of each eye's orientation.

Fig. 4a illustrates the simplified eye model we use to track gaze. The model represents an eye using two spheres: an eyeball sphere to represent the sclera and center of rotation, and an offsetted sphere to represent the corneal surface. The model consists of five per-sequence parameters: eyeball sphere center $\mathbf{c}_e \in \mathbb{R}^3$ relative to the head, iris depth d_i, iris radius r_i, corneal sphere depth d_c, and angle $\boldsymbol{\kappa} \in \mathbb{R}^3$. $\boldsymbol{\kappa}$ represents the angle between the optical axis and the visual axis (i.e., the direction a person looks in when they fixate on a point in space), which we parameterize as an axis-angle rotation vector. Additionally, for each frame, we parameterize gaze direction as the unit vector parallel to the visual axis, and we concatenate left and right eye gaze directions into a single vector $\mathbf{g} \in \mathbb{R}^6$. To build a

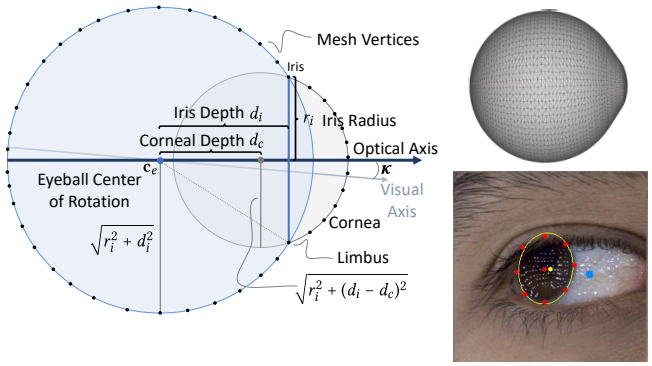


Fig. 4. **Eyeball Geometry Model.** The geometry is defined as the surface formed by the union of two spheres, linearly blended around the boundary. The model parameters, shown above, fully define the shape and are fit using multi-view iris keypoint detections. For later use as cross-domain correspondence, we estimate the visual axis using LED fixation targets.

3D triangle mesh of the eye, we blend the corresponding vertices of each sphere together based on the signed angle from sphere center to the iris contour using a smoothstep function (see A for details).

To fit this model to the data, we project 3D points on the iris contour into the image and minimize the distance to the 2D keypoint detections. We estimate the visual axis using 9 LED fixation targets with known positions. Fig. 5 shows an example of the quality of the fitting process and corresponding keypoint detections. Note that this fitting only works when keypoints are detected in the image. We will discuss in §5.2.2 how we automatically deal with cases where the keypoint detector fails, including when the iris is occluded by the eyelids.

Although the true geometry and motion of the eye is very complex (as shown in Bérard et al. [2014]), we use the simple model described in Fig. 4a for two important reasons. First, the geometry is fully parametric and can be produced in a differentiable fashion. This is important for learning, as we will eventually refine these estimates using differentiable rendering (§5.2.2). Second, the geometry can

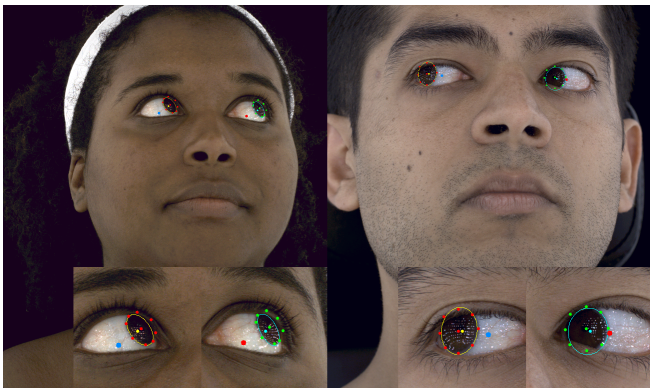


Fig. 5. **Eyeball Fitting Results.** Shown are iris keypoints used for fitting, along with the reprojected limbus contour (the circle formed by the intersection of the eyeball and cornea sphere surface) and the center of the eyeball sphere (blue/red points). Given sufficient views, orientation estimates are robust to noise in keypoint predictions.

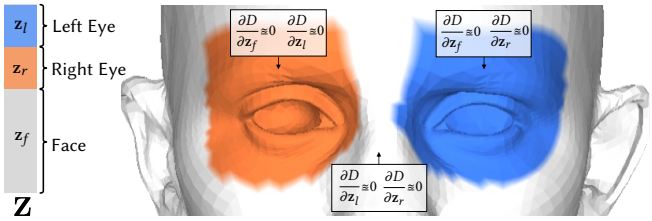


Fig. 6. **Latent Space Partitioning.** To ensure proper control of the model's apparent gaze directions at all vergence distances, the model must allow for independent control of each eye. With no additional losses, however, the model will take advantage of correlations in the training data to minimize the capacity dedicated to independent eye appearance. We avoid this by breaking the latent space into segments for the left eye, right eye, and face, and penalizing the gradient of unrelated facial areas w.r.t. each segment (i.e., changing the left eye latent code or gaze does not change the right side of the face).

be produced using only simple operations and has relatively few polygons. This is important if we are to render the eyes in a real-time system, especially if we need to render at the high framerates used in modern VR displays.

4.2 Gaze-Conditioning in Deep Appearance Models

A straightforward approach to allow independent control of gaze and expression in a deep appearance model would be to provide gaze direction as an explicit gaze conditioning input $\mathbf{g}$, to the decoder network (in addition to the expression code), with $\mathbf{g} \in \mathbb{R}^6$ the concatenated left and right gaze directions.

Simply using gaze direction estimates as an additional conditioning input, however, is not sufficient to create a model that offers direct gaze control. It has been shown that merely providing additional inputs to a VAE-based model does not guarantee that they will be used [Higgins et al. 2017]. If the VAE encoder is able to infer the content of the conditioning input, it may place that content in the latent code $\mathbf{z}$ and the conditioning input may be ignored by the decoder. This process of entangling signals in the latent space has been studied in the past, and various methods have been proposed to learn disentangled representations (e.g., FaderNets [Lample et al. 2017] and MINE [Belghazi et al. 2018]).

We use an approach inspired by FaderNets, wherein we generalize their adversarial discriminator and classifier to the continuous case,

$$\ell_{\text{dis}} = \|\mathbf{g} - C(\mathbf{z}|\theta_{\text{dis}})\|_2^2 \tag{1}$$

$$\mathbf{z} \sim E_f\left(\bar{\mathbf{G}_f}, \bar{\mathbf{T}_f^a}|\theta_f\right), \tag{2}$$

where C is a 2-layer MLP with input $\mathbf{z}$, trained adversarially to minimize ℓ_{dis} w.r.t. parameters θ_{dis}, while maximizing ℓ_{dis} w.r.t. the VAE encoder E_f with parameters θ_f, which takes tracked geometry $\bar{\mathbf{G}_f}$ and average texture $\bar{\mathbf{T}_f^a}$ as inputs and samples $\mathbf{z}$ (see Fig. 3).

Adding a disentangling loss allows us to drive the apparent gaze of the model independent from the expression, but unfortunately, this control comes at the cost of reconstruction quality as we see in Fig. 2a and quantitatively in §7.1. Furthermore, and perhaps more important for eye-contact, Fig.2b highlights a particular form of overfitting present in the model: the eyes cannot converge at any other distance than the one in the training data. To be precise, the model's eyes converge roughly at 1 meter from the face, which corresponds to the radius of the capture system used in Lombardi et al. [2018]—indeed, everything in the subject's field of view during a data capture lies at this distance. To achieve eye contact, or indeed vergence at any other distance, we must therefore overcome an important limitation of deep appearance models: they tend not to generalize to situations that do not occur in the training data.

4.3 Region Separation

As we saw in Fig. 2b, even with direct gaze control, if we provide a novel combination of previously-seen left/right gaze directions (such as a new vergence depth), the model is unable to reproduce it. Since we only provide training examples at a fixed vergence depth, the left and right gaze signals are highly redundant. We could remedy this by collecting training examples at multiple vergence depths but this would add yet another axis to the already large data collection space.

For maximum generalization ability, we want a model that respects the causal independence of the left and right eyes. While they are naturally correlated, their movements are physically independent [Dell'Osso 1994]: when we give a signal to move one eye, the other should not move. We can conceive of a model that achieves this property by behaving in the following way: the left eye region changes only if the left eye gaze signal changes, and likewise for the right. Formally, we do this by adding an additional loss term that minimizes the gradient of the output of the model everywhere but the left eye region with respect to the left gaze signal (and likewise for the right). Assume a decoder for facial appearance D_f with parameters θ_f, inputs $\mathbf{g} \in \mathbb{R}^6$ containing left and right gaze orientation vectors, view vector $\mathbf{v} \in \mathbb{R}^3$, and latent code $\mathbf{z} \in \mathbb{R}^{256}$ describing the state of the face, and producing a face texture $\mathbf{T}_f$ and geometry $\mathbf{G}_f$:

$$D_f\left(\mathbf{z}, \mathbf{v}, \mathbf{g}|\theta_f\right) \rightarrow \mathbf{G}_f, \mathbf{T}_f, \tag{3}$$

and we have a binary indicator mask $\mathbf{M}_{\widetilde{\text{left}}}$ that is 1 everywhere *except* for the left eye region, conceptually we'd like to enforce

$$\frac{\partial\left(\mathbf{M}_{\widetilde{\text{left}}} \odot D_f\left(\mathbf{z}, \mathbf{v}, \mathbf{g} | \theta_f\right)\right)}{\partial\{\mathbf{g}_{\text{left}}, \mathbf{z}_{\text{left}}\}} \approxeq 0, \tag{4}$$

in the case of the left eye and likewise for the right eye. Note that, although the explicit gaze signals $\mathbf{g}$ encode eye orientation, the latent code $\mathbf{z}$ still contains the orientation-independent eye state (e.g., blinking, general eyelid-openness). Therefore, we apply the same penalty to gradients w.r.t. $\mathbf{z}$, but since the latent space is not semantically divided like the geometry, texture, and gaze, we assign fixed-size blocks of the latent code to correspond to orientation-independent left eye state $\mathbf{z}_{\text{left}}$ and right eye state $\mathbf{z}_{\text{right}}$.[1] Fig. 6 shows the regions of geometry and texture corresponding to each eye.

For the sake of completeness, we should theoretically also minimize the complement of the above loss, namely, minimize the gradient of the left-eye-specific outputs w.r.t. everything that doesn't control the left eye region, but in practice one of these two losses is sufficient. For efficiency, rather than directly computing this gradient loss with an additional backward pass, we approximate it with finite differences by scrambling the other blocks of the latent space and minimizing the difference between the non-scrambled and scrambled output in the region of interest:

$$\ell_{\text{left}} = \left\| \mathbf{M}_{\widetilde{\text{left}}} \odot \left(D_f\left(\mathbf{z}, \mathbf{v}, \mathbf{g} | \theta_f\right) - D_f\left(\mathbf{z}_{\text{per}}, \mathbf{v}, \mathbf{g}_{\text{per}} | \theta_f\right)\right)\right\|_2^2 \tag{5}$$

$$\mathbf{z}_{\text{per}} = \left[\mathbf{z}_{\text{left}}, s(\mathbf{z}_{\text{right}}), s(\mathbf{z}_{\text{face}})\right] \tag{6}$$

$$\mathbf{g}_{\text{per}} = \left[\mathbf{g}_{\text{left}}, s(\mathbf{g}_{\text{right}})\right], \tag{7}$$

where $s(\mathbf{z})$ is a function which scrambles its inputs[2].

Following the above process, we can indeed (as shown in §7.2) build a model that offers direct control over the apparent gaze via gaze-conditioning and disentangling, as well as independent left and right eye control for novel vergence. Unfortunately the trend of trading generalization for reconstruction error continues, and, as we show in §7.1, this model has the worst training reconstruction error so far. Additionally, we have been working so far with the facial geometry model used in Lombardi et al. [2018], which covers the eyelids with flat polygons since they did not track eyes. If the face geometry does not model the eyeball surface then, from side views, the eyeball appears truncated (Fig. 2c). The model could potentially draw the remainder of the eyeball on the skin behind it, but generalization to new viewpoints is poor when the geometry is incorrect, as originally shown in Lombardi et al. [2018].

5 JOINT EYEBALL AND FACE MODEL

Many of the failure cases discussed above stem from the low-resolution geometry used by previous methods. This geometry was chosen due to the ease with which it can be tracked and registered. The downside is that when this geometry is inaccurate, large distortions in texture are required to match observed views. These distortions do not generalize, as we have seen above.

We will show that, even if we cannot track and reconstruct the periocular region as precisely as the rest of the face, a rough tracking, combined with geometry and texture which are both generated differentiably, can greatly improve fidelity. The model of Bérard et al. [2016] would be a promising candidate, but as-proposed, it is not amenable to the gradient-based optimization we would like to perform. We could also consider even more complex models such as a full eye rig with wetness layers and refraction. Our goal, however, is to learn facial appearance models in a fully autonomous, scalable fashion.

5.1 Learning-Amenable Eyeball Model

We instead propose a process, described in Fig. 3, to jointly learn a model of facial and periocular appearance, represented by geometry and view-dependent texture. One important component in this process is a model for eyeball appearance that is amenable to image-based learning while still respecting the underlying geometric properties of the human eye as much as possible. Conveniently, the simple geometric model used for gaze tracking in §4.1 is well-suited for the task of rendering eye appearance, when paired with an appropriate view-dependent texture.

Rather than take a VAE-based approach, as in Lombardi et al. [2018], we produce view-dependent, gaze-dependent eyeball textures using a minimal amount of inputs, all of which have semantic meaning. Specifically, we produce the textures using a decoder-only architecture (Fig. 7) which takes gaze, viewpoint, and eyelid shape (represented as vertex positions along the eyelid boundary) as inputs. Gaze and view are necessary to model specular reflection and refraction. The eyelid shape is necessary to model all of the challenging effects that appear along the interface between the eyeball and the face, including wetness and ambient occlusion. Formally, the full eyeball geometry $\mathbf{G}_e$ and texture $\mathbf{T}_e$ are formed by a single

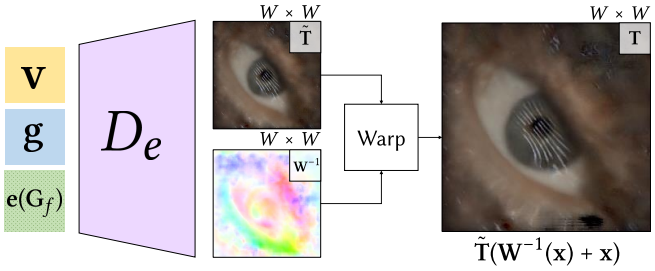


Fig. 7. **Eyeball Texture Decoder.** We produce per-eyeball view-dependent textures via a decoder architecture that takes gaze direction, eyelid shape, and viewpoint as inputs. Gaze direction and viewpoint are required to model specular effects such as glints and refraction. Eyelid shape defines shading and occlusion effects. The decoder is a series of transposed convolutions producing a final texture and associated warp field [Shu et al. 2018]. For all experiments, $W = 256$.

[1] In our experiments, the first 32 components control the left eye, the second 32 components the right, and the remaining 192 control the rest of the face, for a latent space with 256 dimensions.

[2] Similar to MINE, we simply permute entries along the batch axis.

model,

$$D_e\left(\mathbf{g}, \mathbf{v}, \mathbf{e}(\mathbf{G}_f)|\theta_e\right) \rightarrow \mathbf{T}_e, \mathbf{G}_e, \tag{8}$$

where $\mathbf{e}(\mathbf{G}_f)$ extracts the vertex positions of the eyelids using a pre-defined mask.

To better model details like glints, we decode both a texture and a warp field similar to that of Shu et al. [2018]. The warp field is applied to the decoded texture to produce the final view-dependent texture. We also add a similar warp field as a component of the face texture decoder.

Unfortunately, the process described in §4.1 does not estimate cornea geometry, since it relies only on iris keypoints. In Bérard et al. [2016], they fit a model derived from an existing database of eyeball geometries and iris textures ([Bérard et al. 2014]) to the sclera segment of 3D facial reconstructions, using texture synthesis to produce an iris texture. Such a method produces accurate fits when the 3D reconstruction is of high quality. In a fully autonomous large-scale pipeline, however, we cannot rely on having a high-quality 3D reconstruction of the sclera at all times and for all subjects.

We address the issue of model fitting, as well as other challenges that arise in the process, using a joint eye and face learning framework based on differentiable rendering.

5.2 Eye-and-Face Learning Pipeline

The eyeball model described in §4.1 can provide a rough initial estimate for eyeball geometry, but we must refine those estimates and also produce textures if we are to render the eyeballs in a face. Given a multi-view facial performance capture, we estimate eyeball orientations for each frame, along with initial shape parameters for the model described in §4.1. Using these estimates, we learn a set of latent codes describing the gaze-independent state of the face (§4.2, §4.3). From these fixed per-frame codes, we decode face geometry and texture using a model based on the deep appearance model of Lombardi et al. [2018], decode eye texture using the model described in §5.1, render the result, and optimize all decoders to match images. In the following sections, we refer to this model as an **Explicit Eyeball Model (EEM)** to highlight the fact that the eyeball is a separate, directly-controlled geometric component rather than an implicit component modeled entirely by view-and-expression-dependent texture.

5.2.1 Defining a Latent Space. We showed various failure cases of deep appearance models in §3 when modeling the periocular region, but we still need a facial mesh whose shape agrees with the current eye state, and the losses in §4.2 and §4.3 do produce a latent space with independent control over the left and right eye regions of the face (despite reducing fidelity). Since we are improving the geometry and texture in these regions, however, we incorporate the losses of §3 into a preprocessing step that produces a latent space as a byproduct. We can use this latent space later to control the face that surrounds the added eye geometry, and improve said geometry via differentiable rendering.

This latent space step consists of training a Deep Appearance Model VAE following the method of Lombardi et al. [2018], with gaze conditioning, a disentangling loss, and a region separation loss. For a given frame and camera viewpoint, the full loss we optimize in the latent space creation step is:

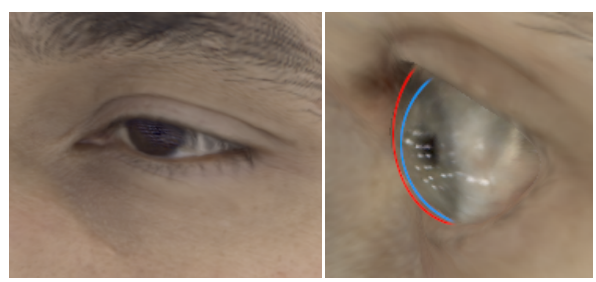
$$\mathcal{L} = \left\|\bar{\mathbf{T}}_f - \mathbf{T}_f\right\|_2^2 + \left\|\bar{\mathbf{G}}_f - \mathbf{G}_f\right\|_2^2 + \ell_{\text{left}} + \ell_{\text{right}} + \ell_{\text{dis}} + \ell_{\text{KL}}, \tag{9}$$

where ℓ_{KL} is the standard KL-divergence loss in a variational autoencoder, and $\bar{\mathbf{G}}_f$, $\bar{\mathbf{T}}_f$ are tracked geometry and texture. After this learning step, we have low-dimensional encoding vectors $\mathbf{z}$ for all frames in our training dataset. We drop the learned encoder E_f and use these encodings as a fixed representation of facial state from which we may decode the various geometries and textures we will need later.

While we separate latent space creation and facial appearance modeling into two stages, it is conceivable that such a process could be performed in a single end-to-end step. Such a process would likely bring many additional challenges, however, and we leave this optimization for future work.

5.2.2 The Eye-Face Interface. Given a simple eyeball model and associated orientation estimates, a straightforward approach would be to simply train a deep appearance model by unwrapping eye textures along with face textures, as in Lombardi et al. [2018]). Unfortunately, our gaze estimates are not always correct. Furthermore, there are situations where estimating gaze is difficult or impossible, such as blinking or closed eyes. If we simply discard samples where we failed to estimate gaze, which we must do if we follow the straightforward approach, we will be unable to reproduce certain gaze directions or expressions which cause missing gaze (e.g., blinking), as we show in Fig 8.

Existing models for facial appearance, particularly in the periocular region, model the appearance in one of two ways: a machine-learning-based model with implicit controls (i.e., a latent space), or a set of textured surfaces (skin, eyeball, wetness, iris state, etc.) with explicit controls (i.e., a traditional facial rig). Our proposed method lies somewhere between these two extremes. We have a rough estimate for eyeball and face geometry, but the estimates are not precise



(a) Missing Gaze (b) Incorrect Cornea

Fig. 8. **The Need for Differentiable Rendering.** In (a), all blinking frames had no gaze estimates, thus blinking was effectively removed from the training set and the decoder cannot produce a blinking expression. With differentiable rendering, we can jointly optimize eye geometry and texture for these frames to match images without initial gaze estimates. In (b), we see the initial position of the cornea in red vs. the true position in blue. We do not estimate this during the gaze estimation process, but differentiable rendering allows us to optimize this parameter to match images.

enough to act as a traditional facial rig, and are not complete enough to be used as a straightforward replacement for a deep appearance model. If we simply combined a facial mesh obtained from 3D reconstruction with the tracked eyeball, the resulting appearance would be incomplete and difficult to correct.

To handle cases where we have missing or imprecise gaze estimates, and simultaneously solve the issues that arise when integrating an explicit eyeball with the face, we propose to replace reconstruction losses on tracked geometry and texture with an image-based loss which we optimize via differentiable rendering. This allows the mesh and texture of the face and eyeball to deviate from tracked results where necessary to fully explain the observed images. In particular, this allows us to model expressions where eyes are partially or fully occluded, and the resulting interface between eyes and face is no longer visible.

Using decoders for the eyes and face, we produce corresponding textures and geometry. For the face, its geometry is also produced by the face decoder. We remove the triangles covering the eyeball in the original model since we now have proper geometry to fill that region. For the eyeball geometry, we simply rotate the eyeball model to the orientation specified by the gaze input[3]. We then rasterize the two meshes and render the result differentiably to get an image. We optimize the difference between rendered and ground-truth images w.r.t. decoder parameters. Formally, given face and eye decoders $D_f\left(\mathbf{z}, \mathbf{v}, \mathbf{g}|\theta_f\right), D_e\left(\mathbf{g}, \mathbf{v}, \mathbf{e}(\mathbf{G}_f|\theta_e)\right)$, ground-truth image $\mathbf{I}$ and a differentiable renderer $R(\mathbf{G}, \mathbf{T}, \mathbf{p})$ rendering from camera parameters $\mathbf{p}$, we optimize the following:

$$\arg\min_{\theta_f, \theta_e} \sum_i \left\|\mathbf{I}_i - \hat{\mathbf{I}}_i\right\|_2^2 \tag{10}$$

$$\hat{\mathbf{I}}_i = R\left(\left[\mathbf{G}_e^i, \mathbf{G}_f^i\right], \left[\mathbf{T}_e^i, \mathbf{T}_f^i\right], \mathbf{p}_i\right),$$

where $i \in \mathcal{I}$ denotes the set of training images from the multiview capture system.

There are a number of existing methods for differentiably rendering a textured triangle mesh, each with different tradeoffs and optimization properties. In this work we use a simple strategy whereby for each rendered triangle, we extrapolate the barycentric coordinates outside the edges and use these as blending weights to average shifted versions of the render. Since these blending weights are differentiable functions of the input vertices, they allow gradients to propagate back to the vertices across silhouette edges and depth discontinuities. This is similar to the approach of Liu et al. [2019], however, we are not concerned with soft occlusions or other transparency effects.

Ideally, we would produce the exact "ground-truth" geometry with our model, and view-dependent texture would only explain true view-dependent effects like specularity rather than inconsistencies in the geometry. Unfortunately, we do not have and cannot expect perfect 3D reconstruction for the entire face. If we simply leave geometry and texture unconstrained so that they can best explain the images, however, the initial geometry is so far from the true shape that image-based losses cannot provide useful signal. To solve this issue, we regularize the decoded geometry to match the coarse tracked facial mesh of Lombardi et al. [2018] with a weight that decays over the course of training[4].

5.2.3 Handling Missing Gaze Data. Learning a facial appearance model using differentiable rendering as described above introduces an ambiguity: any given image of the eye can be explained by an eyeball mesh oriented properly, drawn with the correct texture, or it could be explained by an eyeball with the wrong orientation, with a texture that has been shifted to compensate. For frames where we have accurate gaze estimates, this is not an issue as the eyeball is properly oriented and will thus receive the correct texture. On frames where we have missing or inaccurate gaze estimates, however, the texture will simply compensate by drawing the iris in the wrong place. This will require large distortions in the texture which we saw in §4 do not generalize. This ambiguity between dynamic geometry and dynamic texture is similar to ones seen in shape-from-shading and photometric stereo methods where multiple possible combinations of albedo and surface geometry must be resolved via regularization or other techniques.

To address this, we impute missing gaze estimates using a regressor network which maps from the ground-truth texture to a gaze direction. For frames where we have gaze estimates, we penalize the difference between the regressor output and the gaze estimate. Otherwise, the only signal is the image loss. This regressor quickly learns to predict gaze directions for frames with no estimate, allowing the texture to model only what it needs to. This regressor is only used to fill in gaps during training, after which we discard it.

6 DRIVING EEMS IN REAL-TIME

By introducing an eyeball model that is well-suited for differentiable rendering, and carefully balancing the learning process for unknown or inaccurate gaze directions, we are able to produce a factorized eye and face model that reproduces the eye appearance with high fidelity. The challenge now is driving such a model in real-time.

Wei et al. [2019] demonstrated that it is possible to drive avatar models in real-time only from headset-mounted cameras (HMC). In our case, however, we now require the encoder to predict gaze directions as an additional input to our face model with explicit eyeballs. In the following, we first introduce an additional gaze data collection in HMCs to provide additional gaze labels (§6.1), a novel algorithm for establishing correspondences between HMC images and avatar facial expression and gaze direction (§6.2), and finally, the real-time architecture for driving avatars (§6.3).

6.1 HMC Gaze Data Collection

To establish precise correspondences, Wei et al. [2019] introduced the idea of first using a "training headset" with 9 additional cameras to provide additional multi-view supervision, and then building a real-time model that uses only images from a subset of 3 cameras. Later, a "tracking headset" with only those 3 cameras can be used for real-time animation. We follow this approach and their data collection process to collect facial expression data. To drive our EEM with better precision, however, we add explicit labels for a

[3]We also estimate torsion during training, but not when driving an avatar in real-time.

[4]Specifically, with a schedule of initial_weight $\cdot (1-t)^2$ + final_weight where $t = \max\left(1 - \frac{\text{iter}}{\text{full\_decay\_iter}}, 0\right)$.

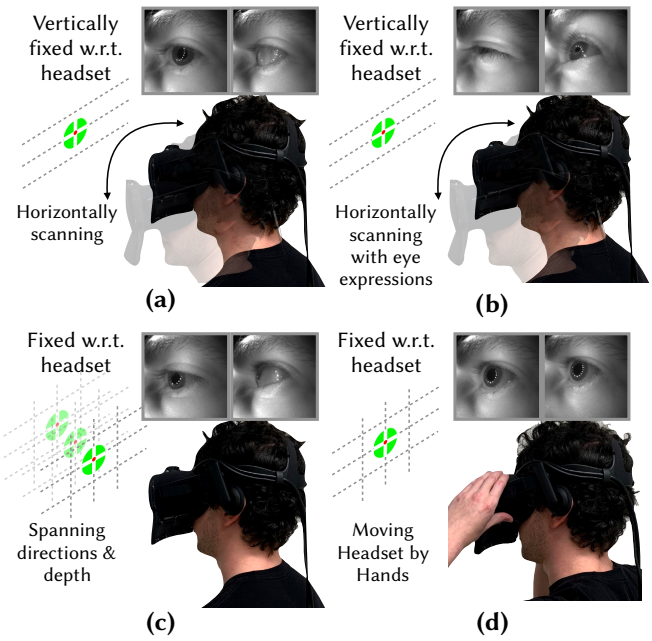


Fig. 9. **HMC Gaze Data Collection: (a)** The target is locked on a horizontal line s.t. the vertical level is fixed relative to the headset, while the horizontal position is fixed in virtual space. This means that rotating the subject's head while fixating on the target causes their eye to rotate only horizontally. Subjects are instructed to rotate their head to produce continuous horizontal gaze variation. The process is repeated with lines at different vertical levels. **(b)** Same as (a) but subjects are instructed to make expressions such as widely-open eyes and squinting. The background intensity in the VR scene is also changing, so that we can collect variations in pupil size. **(c)** Targets are fixed relative to the head and appear at various directions and depth, so resulted data are suitable for evaluation. **(d)** To simulate different HMC wearing positions, subjects are instructed to move the headset around while staring at the target like (c).

number of gaze directions. Using analysis-by-synthesis alone, the domain gap between HMC images and rendered avatars hinders precise alignment of the eyeball geometry (see §7.3.2).

In practice, it is difficult to collect data spanning multiple expressions while also getting clean gaze direction labels for every frame. In this work, we separately collect two sessions of HMC data: one with various expressions (with no constraints on gaze direction), and another where the user is asked to strictly stare at targets at different positions while changing eye expressions (but with almost no lower face variation). Figure 9 illustrates our design of 4 different sections of gaze data collection, which aims to cover the full span of gaze directions, fixation depths, eye expressions, and HMC wearing positions, while being consistent in the distribution of collected gaze labels across different subjects. Our design is driven by the observation that gaze label noise is much lower if the relative movement of the target to the subject's head is controlled by the subject's head movement, rather than have the subject follow a moving target (in this way, we take advantage of the vestibulo-ocular reflex [Fetter 2007]).

6.2 Establishing Correspondences

To drive our EEM in real-time using a virtual reality headset, we propose a method inspired by the analysis-by-synthesis pipeline presented in Wei et al. [2019]. The overall system is illustrated in Fig. 10. Given synchronized multiview images $\mathcal{H} = \{\mathbf{H}_c\}_{c \in \mathcal{C}}$ captured from a set of headset-mounted cameras $\mathcal{C}$, our goal is to estimate facial expression, and additionally, gaze directions in the virtual space with respect to the headset ("headset space").

Specifically, we estimate the parameters ϕ of a regressor $\mathbf{E}_\phi$ that extracts $\{\mathbf{z}^t, \mathbf{p}^t, \mathbf{g}^t\}$, the latent code, the avatar's pose with respect to HMC, and gaze directions for frame $t \in \mathcal{T}$, by jointly considering data from all cameras:

$$E_\phi(\mathcal{H}^t) \rightarrow \mathbf{z}^t, \mathbf{p}^t, \mathbf{g}_0^t, \ \forall t \in \mathcal{T}. \tag{11}$$

Here, $\mathbf{g}_0 \in \mathbb{R}^4$ is in the headset space used during capture, and parameterized as horizontal and vertical angles of both eyes. To convert it into $\mathbf{g} \in \mathbb{R}^6$ in the avatar's coordinate frame, we additionally estimate 6 parameters for the unknown rigid transformation $\mathbf{W}$ from headset space to the HMC camera space, before using predicted $\mathbf{p}^t$ to transform from reference HMC camera's space to avatar's space:

$$\mathbf{g}^t = \mathbf{p}^t(\mathbf{W}(\mathbf{g}_0^t)). \tag{12}$$

Unlike Wei et al. [2019] relying on training separate generative adversarial networks (GANs) to mitigate the domain gap between HMC images and avatars, we instead jointly train shallow fully convolutional networks G_ψ that transform the avatar's view-dependent face texture $\mathbf{T}_f$ and eyeball textures $\mathbf{T}_e$ to HMC-like textures $\mathbf{U}_f$ and $\mathbf{U}_e$ before we render them from the perspective of HMC cameras (for brevity we hide all t superscripts from now on). Specifically, we decode geometry $\mathbf{G}$ and view-dependent textures, and convert their style $|C|$ times, once for every HMC view.

It is important to note that G_ψ is a shallow network, with a small receptive field, operating on high resolution textures. It primarily changes the style of the texture, with almost no ability to alter spatial structures. Because of this property, we can jointly learn E_ϕ and G_ψ, without them compensating the error each other makes to minimize the reconstruction loss but generating incorrect correspondences.

We render these HMC-like textures using the differentiable renderer R to obtain reconstructed images $\hat{\mathbf{H}}_c$,

$$R\left(\mathbf{G}, \mathbf{U}_f^c, \mathbf{U}_e^c, A_c(\mathbf{p})\right) \rightarrow \hat{\mathbf{H}}_c, \ \forall c \in \mathcal{C}. \tag{13}$$

We also learn a per-camera per-pixel linear transform $l_c(\cdot)$ to account for the fixed structures in image space, such as the headset itself and constant shading on the face:

$$\tilde{\mathbf{H}}_c = l_c\left(\hat{\mathbf{H}}_c\right), \ \forall c \in \mathcal{C}. \tag{14}$$

Finally, we define the full image loss function as:

$$L_{\text{image}}(\phi, \psi, \mathbf{W}, l_c) = \sum_{t \in \mathcal{T}} \left(\sum_{c \in \mathcal{C}} \left\| \mathbf{H}_c^t - \tilde{\mathbf{H}}_c^t \right\|_1 + \lambda \delta(\mathbf{z}^t) \right), \tag{15}$$

where δ is a regularization term over the latent codes $\mathbf{z}$, and λ weights its contribution against the reconstruction term. Note that θ of the decoder D is fixed in this process.

Simply optimizing all parameters in an end-to-end fashion with Eq. 15 alone results in two major issues. First, even though G_ψ has

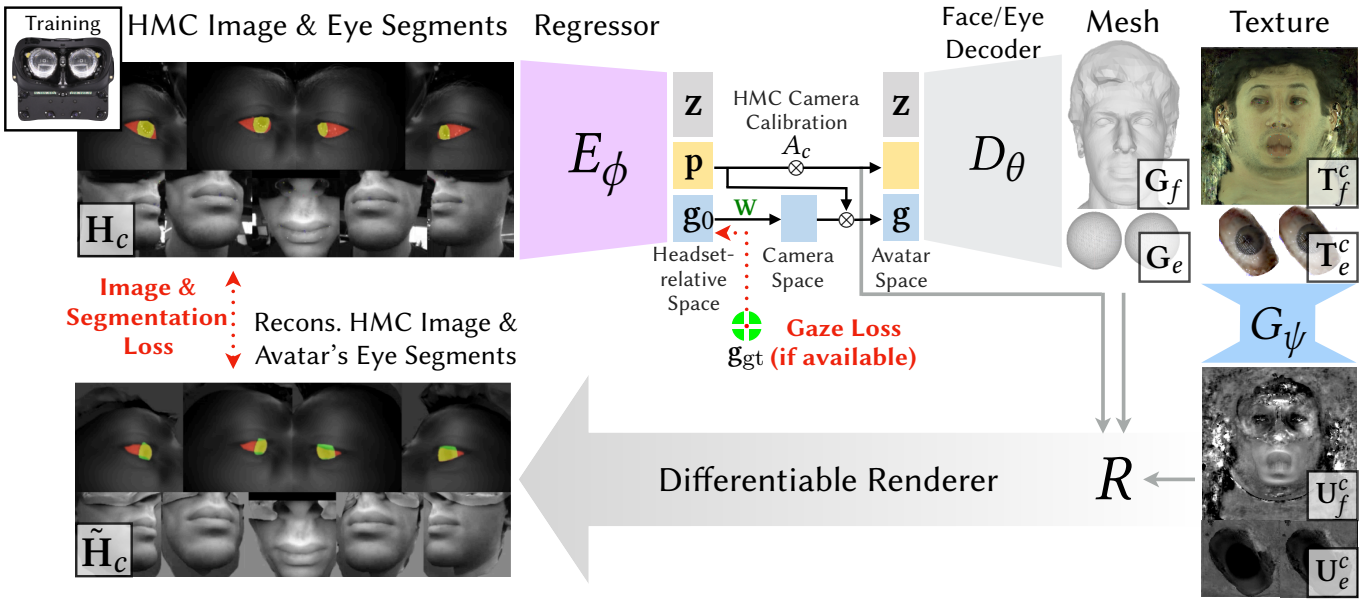


Fig. 10. **HMC ↔ Avatar Correspondence System.** We jointly train networks E_ϕ, G_ψ, unknown rigid transformation $\mathbf{W}$, and per-view linear transformations $l_c(\cdot)$ (not shown in the figure) to minimize reconstructed image loss, eye segmentation (shown in red and yellow) loss, and gaze loss (only for partial dataset with gaze labels).

limited ability to alter spatial structures, small changes in eyeball textures can cause significant gaze error in the correspondence. Even if the eye images appear to be aligned reasonably well from the oblique viewpoints of the HMC cameras, there can still be a high gaze error both numerically and perceptually from the frontal view (not seen by the HMC).

A similar issue caused by small changes in the texture results in the corresponding avatar not fully closing its eyes. Once they are mostly closed, G_ψ may "cheat" and slightly adjust the texture to make them look closed from the view of the HMC.

We introduce two additional losses to prevent the model from making improper changes to the texture during domain transfer. We use 2D eye segmentation, in the form of eyeball and iris segments, $S(\mathbf{H_c})$, formed by detected landmarks:

$$L_{\text{seg}}(\phi, \psi, \mathbf{W}, l_c) = \sum_{t \in \mathcal{T}} \sum_{c \in C_{\text{eye}}} \left\| S(\mathbf{H}_c^t) - S(\mathbf{R}_c^t) \right\|_1, \tag{16}$$

where C_{eye} is the set of eye cameras and $\mathbf{R_c} = R(\mathbf{G}, \mathbf{T}_f^c, \mathbf{T}_e^c, A_c(\mathbf{v}))$ is the rendering of the avatar viewed from HMC camera c. The eye segments $S(\mathbf{R}_c^t)$ can be rendered differentiably by drawing them on our explicit eyeball model. By matching eye segments across domains, correspondences in eye openness and gaze directions are improved.

To further improve accuracy in gaze directions which are not precisely described by 2D segmentation in oblique views, we utilize gaze labels $\mathbf{g}_{\text{gt}} \in \mathbb{R}^4$ collected by the process described in §6.1:

$$L_{\text{gaze}}(\phi) = \sum_{t \in \mathcal{T}_g} \left\| \mathbf{g}_0^t - \mathbf{g}_{\text{gt}}^t \right\|_2^2 + \delta_g(\mathbf{g}_0^t), \tag{17}$$

where $\mathcal{T}_g \subset \mathcal{T}$ is the set of frames with gaze labels, and δ_g is a regularization term enforcing the vertical gaze angle of the eyes to be the same, and preventing from fixating at a point too close to the eyes or even behind the eyeballs. Overall the system is trained with

$$L = L_{\text{image}} + \lambda_1 L_{\text{seg}} + \lambda_2 L_{\text{gaze}}, \tag{18}$$

where λ_1 and λ_2 are weights adjusted heuristically. In §7.3.2 we will show the individual effects of these loss terms.

6.3 Real-time Expression Encoding and Gaze Estimation

After minimizing the loss in Eq. (18), we can apply E_ϕ to all $\mathcal{H}^t$ to obtain per-frame correspondences $\{(\mathcal{H}^t, \mathbf{z}^t, \mathbf{g}^t)\}_{t \in \mathcal{T}}$.

Following Wei et al. [2019], we drop all auxiliary views in $\mathcal{H}^t$, retain the 3 views available in a tracking headset $\tilde{\mathcal{H}}^t = \{\mathbf{H}_i^t\}_{i \in C'}$, and build another encoder $\tilde{E}_{\tilde{\phi}}$ to convert $\tilde{\mathcal{H}}^t$ to target $(\mathbf{z}^t, \mathbf{g}^t)$. Instead of simply regressing from 3 images directly, we build an end-to-end trainable, R-CNN [Girshick 2015] inspired network architecture. From a shared image feature, we predict landmarks for eyes as a supervised side task, and only predict $\mathbf{g}^t$ from a tight crop of the shared feature. This design makes the network generalize better to unseen combinations of gaze directions and expressions. Note that we also separately estimate gaze directions for left and right eyes from individual crops of features. This enables our model to generalize better to different depths of fixation, such as crossed eyes, that are not captured in the dataset.

Specifically, dataset $\mathcal{T} = (\mathcal{T}_g, \mathcal{T}_e)$ has two parts where $\mathcal{T}_g$ has gaze labels $\{(\mathcal{H}^t, \mathbf{z}^t, \mathbf{g}^t, \mathbf{g}_{\text{gt}}^t)\}_{t \in \mathcal{T}_g}$ and $\mathcal{T}_e$ only has estimated gaze $\{(\mathcal{H}^t, \mathbf{z}^t, \mathbf{g}^t\}_{t \in \mathcal{T}_e}$ through §6.2. We use 10K frames with gaze labels. Additionally, we prepare another landmark dataset $\{(\mathcal{H}^t, \mathbf{y}^t)\}_{t \in \mathcal{T}_k}$ where $\mathbf{y}$ is heatmaps with Gaussian peaks centered at annotated

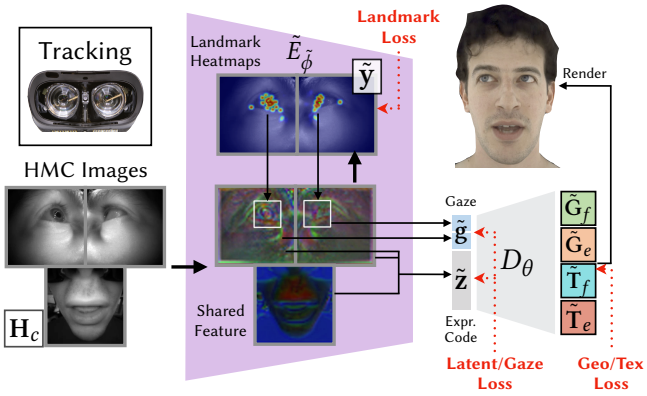


Fig. 11. **Realtime Encoder Architecture.** From 3-view inputs captured by tracking headsets, we first convert them into a shared feature, which is further converted into heatmaps of landmarks. We make tight crops around each pupil center on the shared feature for separate left and right gaze prediction. The expression code is estimated using whole shared feature.

landmarks as learning target, like common landmark detection methods [Wei et al. 2016]. Overall, the loss function for the regressor is a sum of five terms:

$$L_{\text{enc}}(\tilde{\phi}) = L_{\text{latent}} + L_{\text{gaze}} + L_{\text{geo}} + L_{\text{tex}} + L_{\text{kpt}}. \tag{19}$$

The first term is simply L_2 error between predicted and target latent codes:

$$L_{\text{latent}} = \sum_{t \in \mathcal{T}} \left\| \tilde{\mathbf{z}}^t - \mathbf{z}^t \right\|_2^2. \tag{20}$$

The second term is an L_2 error on gaze prediction:

$$L_{\text{gaze}} = \sum_{t \in \mathcal{T}_g} \left\| \tilde{\mathbf{g}}^t - \mathbf{g}^t_{\text{gt}} \right\|_2^2 + \sum_{t \in \mathcal{T}_e} \max\left(0, \left\| \tilde{\mathbf{g}}^t - \mathbf{g}^t \right\|_2^2 - \Delta^2\right), \tag{21}$$

where Δ is a constant margin such that we don't punish gaze estimation $\tilde{\mathbf{g}}^t$ if it's close enough (around 5°, see §7.3.2) to $\mathbf{g}^t$ since for $t \in \mathcal{T}_e$ there is no ground truth gaze. We also penalize the difference between the geometry and texture decoded from our estimated codes vs. the ones decoded from codes obtained in §10. This ensures that even where we cannot precisely match the code via Equation 20, the appearance should still match.

$$L_{\text{geo}} = \sum_{t \in \mathcal{T}} \left\| \tilde{\mathbf{G}}^t_f - \mathbf{G}^t_f \right\|_2^2 + \sum_{t \in \mathcal{T}_g} \left\| \tilde{\mathbf{G}}^t_e - \mathbf{G}^t_e \right\|_2^2 \tag{22}$$

(and likewise for textures L_{tex}). Notice that we don't supervise the geometry and texture of the eyeball for frames without gaze labels, for the same reason as in Eq. (21). Finally, we include a landmark matching loss for annotated iris keypoints:

$$L_{\text{kpt}} = \sum_{t \in \mathcal{T}_k} \left\| \tilde{\mathbf{y}}^t - \mathbf{y}^t \right\|_2^2. \tag{23}$$

Our architecture design balances inference speed with almost no obvious quality drop (from 9-view input to 3-view input) on both training data and validation data, at an inference speed of 50fps on an NVIDIA GTX 1080Ti with 3 views of 192×192 inputs.

7 EXPERIMENTS

We evaluate our full system as well as individual components qualitatively and quantitatively, using the dataset of Lombardi et al. [2018]. For gaze estimation, we additionally train a keypoint detector to predict iris keypoints from annotations, both in images from the headset and images in their capture system.

7.1 Decoder: Quantitative Evaluation

In Table 1, we evaluate the L2 image reconstruction loss in the eye region for a set of models, averaged across 6 diverse subjects. We report losses relative to the baseline model, the deep appearance model of Lombardi et al. [2018], so that they are comparable across subjects. We also compare against the same model with gaze-conditioning (GC), with gaze-conditioning and region separation losses, and finally our full proposed EEM.

As we add losses to improve generalization, the reconstruction error increases for all models except our final proposed method. The addition of gaze-conditioning to the baseline deep appearance model results in increased error likely due to the fact that the model can no longer heavily compress gaze information into the few gaze / expression pairs seen during training. When we add the region loss, the same model must now also disentangle left and right eyes, further reducing its ability to take advantages of spurious correlations in the data. The EEM provides the full benefits of a disentangled representation along with better reconstructions.

Regarding the choice to compare loss within the eye region only: since we learn the EEM eyes and face jointly with differentiable rendering, the result naturally (and perhaps unfairly) matches images better than the indirect geometry and unwrapped texture losses of the baseline methods. Since our primary contributions are eye-centric, we evaluate losses only in the eye region, though a similar comparison on the full face does also show that the full EEM has lower reconstruction error than any baseline.

7.2 Decoder: Qualitative Comparisons

In Fig. 12, we show significant qualitative improvement of the EEM over the baselines, particularly for unseen combinations of expression and gaze. When we provide inputs corresponding to a rare expression (wide-open eyes) and a new gaze direction for that expression (anything besides straight ahead), the baseline degrades significantly. The iris no longer appears circular, the pupil is heavily distorted, and, as can be seen much more clearly in our supplemental video, the perceived gaze direction has been altered. Note that both models in these comparisons receive the same $\mathbf{z}$ and $\mathbf{g}$ inputs.

Fig. 13, shows further novel combinations of expression and gaze, including expressions that modify the face beyond the eyes. Again all are novel combinations of gaze and expression never seen in the training data. As can be seen in our associated videos, the expression can be seamlessly driven independently of the eyes, maintaining quality across all eye poses.

7.3 HMC Correspondences for Realtime Encoding

7.3.1 Comparisons and Qualitative Results of Correspondence. In Fig. 14, we compare established correspondences with the method described in §6.2 with EEMs against Wei et al. [2019] which used

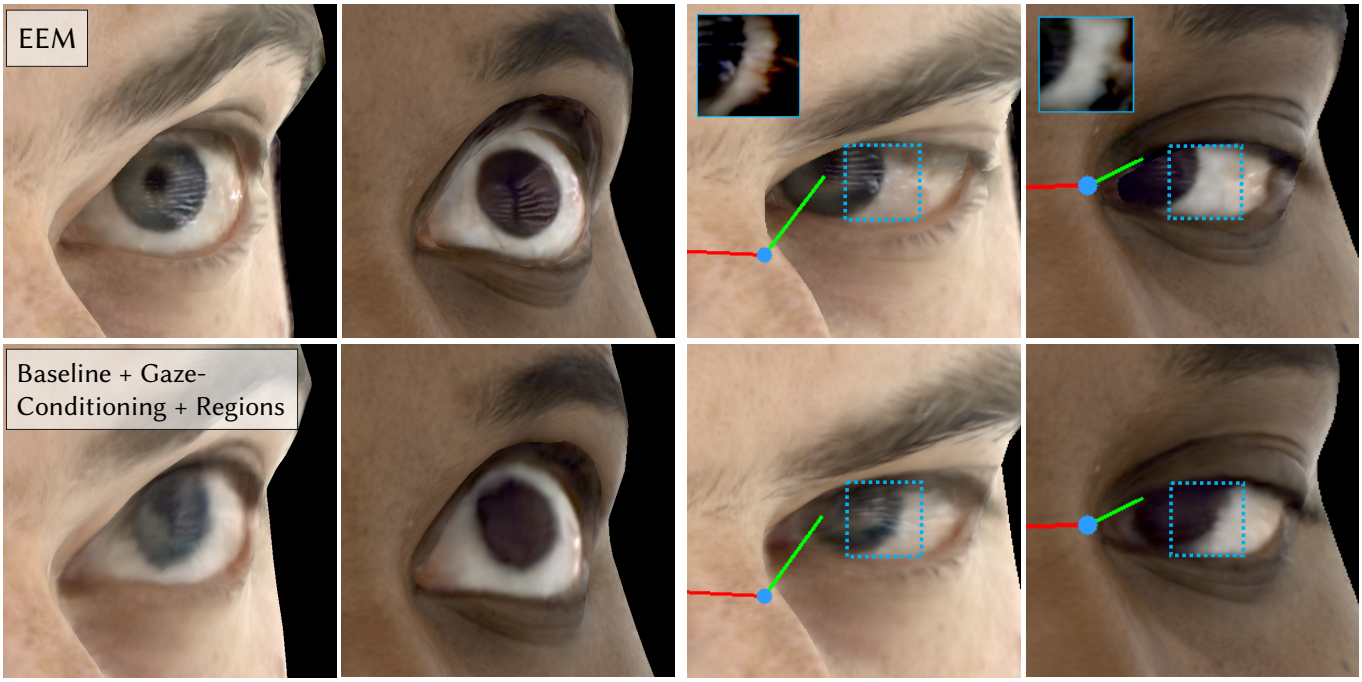


Fig. 12. **Improvements from Explicit Eyeball Modeling on *Unseen Input Combinations*.** These images show the ability of our model to generalize to novel inputs. The first two columns show an unseen combination of a rare expression and new gaze direction. The baseline (regions + gaze-conditioning) distorts the iris and pupil in very uncanny ways. Though the region loss of §4.3 enables independent control, the model generalizes poorly to the unseen inputs in the second two columns. In the inset, we show a difference image between our final model and the baseline. The limbus edge is clearly in the wrong location. It is difficult to see in a static image, but in our supplemental video we show this produces a significant difference in perceived gaze direction. Both models exhibit artifacts under these novel conditions, but our proposed method better-preserves the underlying geometry and perceived gaze signal.

Fig. 13. **Further Explicit Eyeball Model Examples.** All of these images show novel combinations of gaze and expression not present in the training data. Expressions have been seen during training, but all with a single straight-ahead gaze. For examples of completely novel gaze and expression, in real-time, see our supplemental video.

deep appearance models [Lombardi et al. 2018]. In the first and third example, the subjects' extreme eye gaze directions in HMC are rare and often get modified in gaze directions by GANs during image style transfer, leading to misaligned and inaccurate correspondences in Wei et al. [2019]. Also, deep appearance models do not have explicit eyeball geometry for applying further constraints to improve it. Our method, on the contrary, does not suffer from the difficult distribution matching with end-to-end training since there is no discriminator involved. Furthermore, we can improve the alignments by matching eye segments through differentiably rendering them with explicit eyeball models. The second and the third example shows a novel combination in input HMC images between eye expression and mouth expression that is not seen in avatar-building data. Deep appearance model is not able to generalize, giving blurry, shrunk, and lightened irises. On the contrary, our

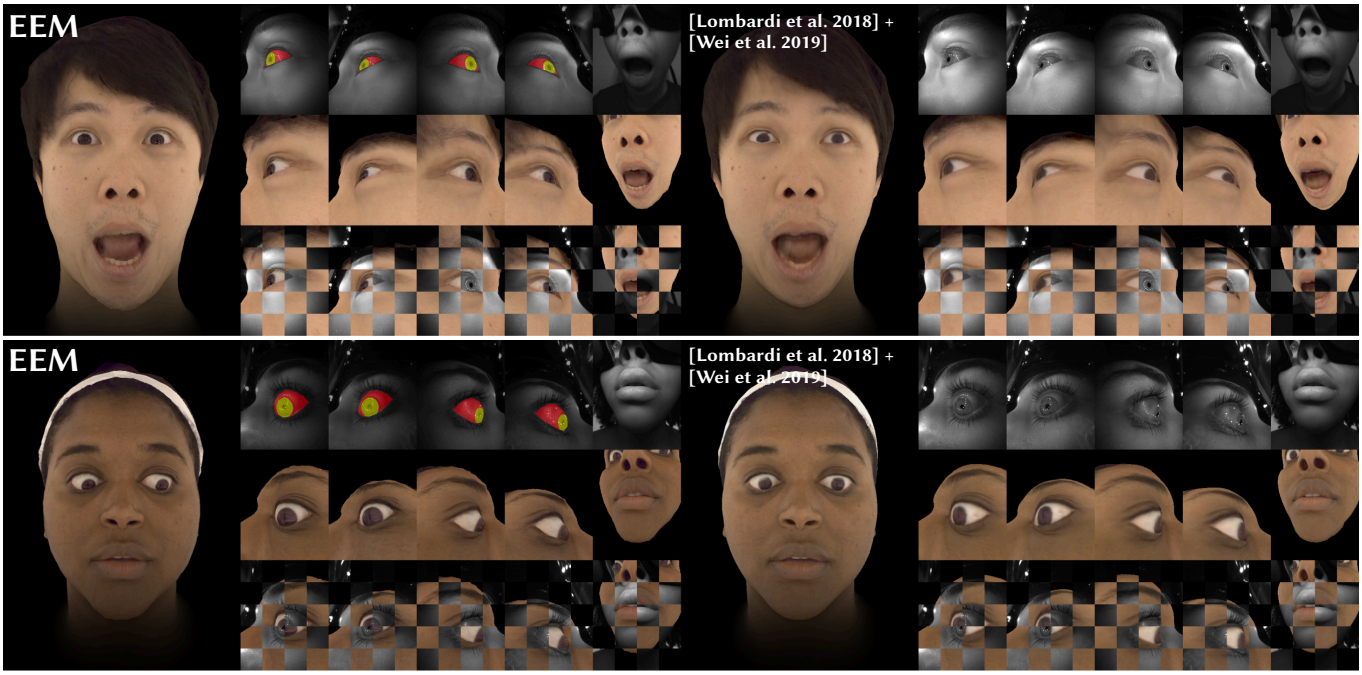


Fig. 14. **Comparison of Established Correspondences.** On the left we show our estimated correspondences between EEM and HMC images. On the right we compare against the method of Wei et al. [2019] applied to deep appearance models [Lombardi et al. 2018]. To the right of each avatar we show, from top to bottom: input HMC images (overlaid with eye segments on the left), the avatar rendered using discovered correspondences from the viewpoint of the headset cameras, and a tiling of the images above to show alignment. Our method aligns eyeballs much better, especially for extreme gaze directions, and it generalizes better to rare combinations of gaze direction and facial expression.

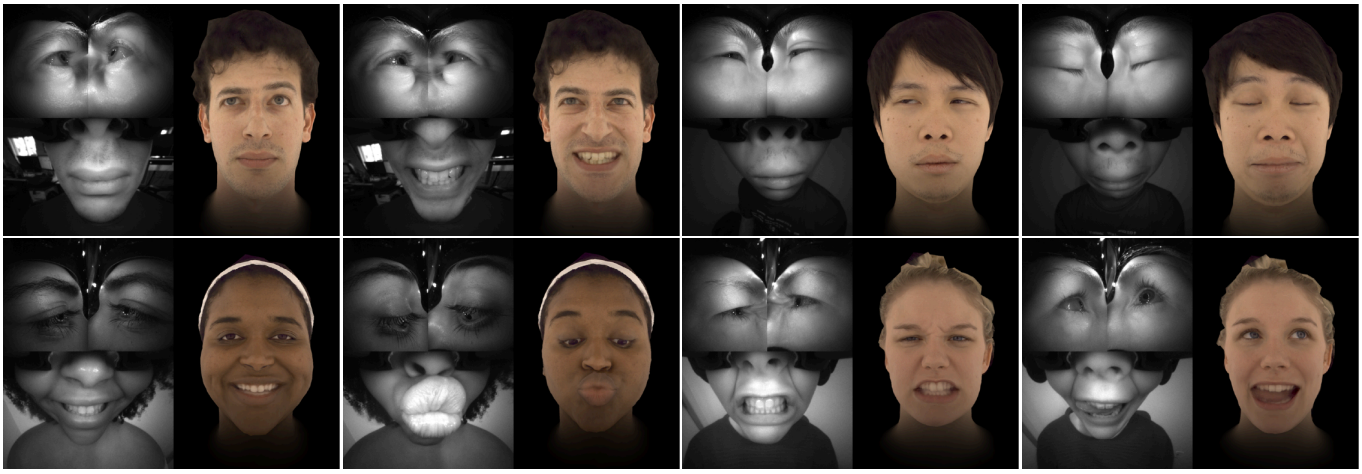


Fig. 15. **Qualitative Correspondences**. Our method generalizes well on various amounts of eye openness, different identities, and combinations of eye and lower face expressions.

method still finds an avatar state that gives realistic irises through EEMs.

Fig. 15 shows more examples of correspondences in different facial expressions, subjects with different eye sizes and shapes, and various eye openness. Our method is able to generalize well on these variations without losing expressivity, thanks to good disentanglement between eye region and other region of the decoder, and the segmentation loss that helps to match eye openness.

Table 1. **L2 Eye Region Reconstruction Loss.** Values are L2 image reconstruction error relative to the baseline deep appearance model, averaged over 6 subjects. These results show that not only is our full EEM perceptually sharper and more consistent, it is also able to better-reconstruct the training data. The addition of gaze-conditioning and region losses to the baseline result in a better-disentangled model at the expense of overall accuracy. EEM provides the full benefit of disentangling as well as improved performance. We should stress that while the reduction in L2 loss may seem subtle, the perceptual improvement is significant.

Model	Rel. L2 Error
Baseline	1.0
+GC	1.085 ± 0.129
+GC +Region	1.097 ± 0.174
EEM	**0.890** ± 0.045

7.3.2 Correspondence Loss Ablation. We introduced three loss terms in Eq. (18), but the individual impact of each loss is not obvious. Table 2 shows an ablation study across two subjects comparing all meaningful combinations of losses. The full system is shown in row A. First, we remove gaze loss[5] in row B and find that there is a significant accuracy drop in gaze in both subjects without obvious change in the other terms. This result shows that image-based features can only provide up to 5° precision in gaze under the oblique views of HMC across the domain gap (and hence the setting of Δ in Eq. (21)). In row C, we further remove segmentation loss and observe a greater drop in gaze accuracy and higher L_{seg}, showing that matching eye segments is indeed preventing G_ψ from changing the structure of the eye texture. In row D, adding the gaze term back leads to low gaze error with higher L_{seg} than row A. This shows that removing L_{seg} leads to overfitting to $\{\mathbf{H}_i\}_{i \in \mathcal{T}_g}$. Row E shows the gaze accuracy of a *held out* set, as a point of reference. We chose to focus on two of the six subjects for this ablation as they had the worst (1) and best (2) validation gaze accuracies in row E.

8 CONCLUSION

In this paper, we investigated the problem of rendering and driving photorealistic models of the human eye and face. While our model offers many improvements over previous methods, there are some known limitations. Our eye appearance generalizes well to novel gaze directions, but if we move away from the space of plausible inputs for a given avatar (i.e. if the gaze input is not physically-realizable), we see artifacts in the resulting rendering. Though we optimize the texture and geometry of our model to match images, this optimization process is not perfect and does rely on good initialization. In cases where the keypoint-based eyeball fitting produces a significantly incorrect shape, it is difficult to recover with differentiable rendering alone. As with any real-time communication system, there is system and network delay. In our supplemental video, there is a ~130ms delay which includes the computation time of the encoder and decoder (~30ms total) and a simulated network delay of 100ms. Finally, our method is only applicable to a single subject and does not yet generalize to multiple identities. While we plan to address this with future work, currently each avatar must be created from captured data for that person.

[5]To be fair, we still use $\mathbf{g}_{\text{gt}}$ to optimize the unknown transformation $\mathbf{W}$.

Table 2. **Ablation of Correspondence Losses**. In the losses column, G means the presence of L_{gaze}, S for L_{seg}, and I for L_{image}. Numbers in rows A-D are training errors, while row E shows errors on a held out test set.

	Losses			Errors (subject 1)			Errors (subject 2)		
	G	S	I	Gaze	L_{image}	L_{seg}	Gaze	L_{image}	L_{seg}
A	✓	✓	✓	0.99°	0.017	0.013	0.69°	0.012	0.007
B		✓	✓	7.76°	0.017	0.013	4.94°	0.013	0.008
C			✓	8.06°	0.017	0.015	9.73°	0.013	0.011
D	✓		✓	0.85°	0.017	0.015	0.75°	0.014	0.009
E	✓			4.08°	-	-	1.10°	-	-

Our primary contribution is a model for the photorealistic appearance of eyes and face that can be driven in real-time. We did this by systematically addressing each of the failure cases discussed in §3. We can obtain direct control over the avatar's apparent gaze direction by introducing gaze-conditioning as an explicit input signal to our model. Introducing disentangling and region separation losses enables the generation of novel combinations of gaze and expression. These novel combinations are unseen in the training data but common in real-world interactions. Our EEM uses these novel gaze / expression combinations, along with tracked eye geometry, optimized jointly via differentiable rendering, to obtain a model that generalizes very well to common face and gaze configurations seen in real interactions. Differentiable rendering minimizes the appearance of artifacts at the boundary of the two surfaces we control. This work represents an important step forward towards creating truly immersive social experiences over any distance, which has the potential to change the way people interact across the world.

REFERENCES

Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. 2009. The Digital Emily Project: Photoreal Facial Modeling and Animation. In *ACM SIGGRAPH 2009 Courses (SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 12, 15 pages. https://doi.org/10.1145/1667239.1667251

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual Information Neural Estimation. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholmsmässan, Stockholm Sweden, 531–540. http://proceedings.mlr.press/v80/belghazi18a.html

Pascal Bérard, Derek Bradley, Markus Gross, and Thabo Beeler. 2016. Lightweight Eye Capture Using a Parametric Model. *ACM Trans. Graph.* 35, 4, Article 117 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925962

Pascal Bérard, Derek Bradley, Maurizio Nitti, Thabo Beeler, and Markus Gross. 2014. High-Quality Capture of Eyes. *ACM Trans. Graph.* 33, 6, Article 223 (Nov. 2014), 12 pages. https://doi.org/10.1145/2661229.2661285

Philippe Bergeron and Pierre Lachapelle. 1985. Controlling facial expressions and body movements in the computer-generated animated short: Tony de peltrie. *Computer Graphics (SIGGRAPH'85), Course Notes: Techniques for Animating Characters* (1985).

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

Milton Chen. 2002. Leveraging the Asymmetric Sensitivity of Eye Contact for Videoconferencing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 49–56.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 2180–2188.

Marvin G. Cline. 1967. The Perception of Where a Person Is Looking. *The American Journal of Psychology* 80, 1 (1967), 41–50.

Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. 1998. Active Appearance Models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Springer, 484–498.

L. F. Dell'Osso. 1994. Evidence suggesting individual ocular motor control of each eye (muscle). *J Vestib Res* 4, 5 (1994), 335–345.

M. Fetter. 2007. Vestibulo-ocular Reflex. *Dev. Opthalmol.* 40 (February 2007), 35–51. https://doi.org/10.1159/000100348

Tobias Fischer, Hyung Jin Chang, and Yiannis Demiris. 2018. RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments. In *IEEE European Conference on Computer Vision (ECCV)*, Vittorio Ferrari, Martial Hebert, Cristiam Sminchisescu, and Yair Weiss (Eds.). 339–357.

Guillaume Francois, Pascal Gautron, Gaspard Breton, and Kadi Bouatouch. 2009. Image-Based Modeling of the Human Eye. *IEEE Transactions on Visualization and Computer Graphics* 15, 5 (Sept. 2009), 815–827. https://doi.org/10.1109/TVCG.2009.24

James J. Gibson and Anne D. Pick. 1963. Perception of Another Person's Looking Behavior. *The American Journal of Psychology* 76, 3 (1963), 386–394.

Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations (ICLR)*.

Tero Karras, Samuli Laine, and Timo Aila. 2018. A Style-Based Generator Architecture for Generative Adversarial Networks. *CoRR* abs/1812.04948 (2018). http://dblp.uni-trier.de/db/journals/corr/corr1812.html#abs-1812-04948

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*.

D. Kononenko, Y. Ganin, D. Sungatullina, and V. Lempitsky. 2018. Photorealistic Monocular Gaze Redirection Using Machine Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 11 (Nov 2018), 2696–2710. https://doi.org/10.1109/TPAMI.2017.2737423

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic DENOYER, and Marc' Aurelio Ranzato. 2017. Fader Networks:Manipulating Images by Sliding Attributes. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5967–5976. http://papers.nips.cc/paper/7178-fader-networksmanipulating-images-by-sliding-attributes.pdf

Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. 2019. Rethinking on Multi-Stage Networks for Human Pose Estimation. *arXiv preprint arXiv:1901.00148* (2019).

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In *IEEE International Conference on Computer Vision (ICCV)*.

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep Appearance Models for Face Rendering. *ACM Trans. Graph.* 37, 4, Article 68 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201401

Matthew Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *IEEE European Conference on Comptuer Vision (ECCV)*.

Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. 2016. High-Fidelity Facial and Speech Animation for VR HMDs. *ACM Trans. Graph.* 35, 6, Article 221 (Nov. 2016), 14 pages. https://doi.org/10.1145/2980179.2980252

Seonwook Park, Shalini De Mello, Pavlo Molchanov, Umar Iqbal, Otmar Hilliges, and Jan Kautz. 2019. Few-Shot Adaptive Gaze Estimation. In *IEEE International Conference on Computer Vision (ICCV)*.

Tom Porter. 1997. Creating lifelike characters in Toy Story. *ACM SIGART Bulletin* 8 (12 1997), 10–14. https://doi.org/10.1145/272874.272876

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1511.06434

Rajeev Ranjan, Shalini De Mello, and Jan Kautz. 2018. Light-weight Head Pose Invariant Gaze Tracking. In *IEEE Computer Vision and Pattern Recognition Workshop (CVPRW)*.

Mike Seymour, Chris Evans, and Kim Libreri. 2017. Meet Mike: Epic Avatars. In *ACM SIGGRAPH 2017 VR Village*. Article 12, 2 pages.

Zhixin Shu, Mihir Sahasrabudhe, Riza Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. 2018. Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance. In *The European Conference on Computer Vision (ECCV)*.

Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality. *arXiv preprint arXiv:1610.03151* (2016).

Tobii VR. 2018. Tobii VR. https://vr.tobii.com/.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *CVPR*.

Shih-En Wei, Jason Saragih, Tomas Simon, Adam W. Harley, Stephen Lombardi, Michal Perdoch, Hypes Alexendar, Dawei Wang, Hernan Badino, and Yaser Sheikh. 2019. VR Facial Animation via Multiview Image Translation. *ACM Trans. Graph.* 38, 4, Article 67 (July 2019), 16 pages. https://doi.org/10.1145/3306346.3323030

Q. Wen, F. Xu, and J. Yong. 2017. Real-Time 3D Eye Performance Reconstruction for RGBD Cameras. *IEEE Transactions on Visualization and Computer Graphics* 23, 12 (2017), 2586–2598.

L. Wolf, Z. Freund, and S. Avidan. 2010. An eye for an eye: A single camera gaze-replacement method. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 817–824. https://doi.org/10.1109/CVPR.2010.5540133

Erroll Wood, Tadas Baltrusaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. A 3D Morphable Eye Region Model for Gaze Estimation, Vol. 9905. 297–313. https://doi.org/10.1007/978-3-319-46448-0_18

A DIFFERENTIABLE COMPUTATION OF EYEBALL GEOMETRY

In this work, we learn physical parameters of the eye, such as eyeball center and radius, by comparing the final rendered output with ground truth images. In order to learn these physical parameters, they must be differentiable with respect to the mesh vertices of the explicit eyeball model. To do this, we construct the mesh vertices in a differentiable way. First, we compute the radius of the eyeball sphere and cornea sphere given the learnable parameters of the model (iris radius, iris depth, and cornea depth):

$$r_{\mathbf{e}} = \sqrt{r_{\mathbf{i}}^2 + d_{\mathbf{i}}^2},$$

$$r_{\mathbf{c}} = \sqrt{r_{\mathbf{i}}^2 + (d_{\mathbf{i}} - d_{\mathbf{c}})^2}.$$

Let $\mathbf{s}$ be the vertices of a triangle mesh of the unit sphere. Next, we compute the angle between the z axis and the edge of the iris,

$$\theta_{\mathbf{i}} = \arcsin r_{\mathbf{i}}/r_{\mathbf{e}},$$

so that we can compute the signed difference between the angle of the edge of the iris and each vertex $\mathbf{s}$,

$$\theta_{\text{diff}} = \arccos \mathbf{s}^z - \theta_{\mathbf{i}}.$$

This angle difference allows us to form a blending function to blend between the vertices of the eyeball sphere and the cornea sphere. First, we compute the blending factor using a smoothstep function:

$$\alpha = \text{smoothstep}(2\theta_{\text{diff}} + 0.5).$$

Next, we form the eyeball vertices by scaling the vertices of the sphere mesh by the eyeball radius,

$$\mathbf{v}_{\mathbf{e}} = \mathbf{s}r_{\mathbf{e}},$$

and form the cornea sphere vertices by scaling and translating the sphere mesh,

$$\mathbf{v}_{\mathbf{c}} = \mathbf{s}r_{\mathbf{c}} + \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} d_{\mathbf{c}}.$$

We blend the vertices of the two spheres using a smoothstep function,

$$\mathbf{v} = \alpha \mathbf{v}_{\mathbf{e}} + (1 - \alpha)\mathbf{v}_{\mathbf{c}}.$$

Then, we rotate the entire eyeball model so that the visual axis points down the z axis:

$$\mathbf{v}_{\text{final}} = \text{Rotate}(\mathbf{v}, \boldsymbol{\kappa}).$$

Now the eyeball mesh is ready to be rotated by the gaze direction and placed inside the head.