ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

Monocular Facial Performance Capture Via Deep Expression Matching

Stephen W. Bailey[1,2] Jérémy Riviere[2] Morten Mikkelsen[2] James F. O'Brien[1]

[1]University of California, Berkeley
[2]Unity Technologies

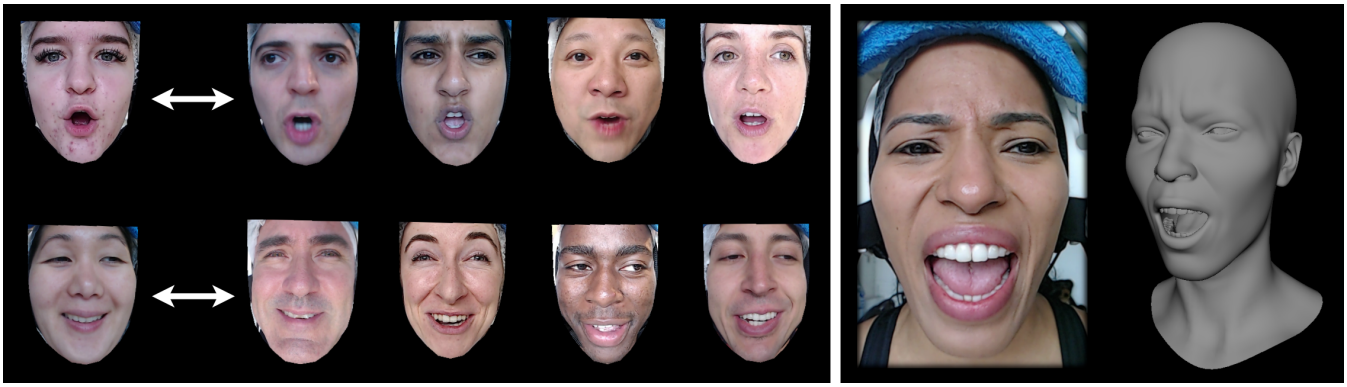


Figure 1: *Examples of expressions matched across recordings of multiple actors (left) and an example of a facial rig posed by our method to match a recorded expression (right).*

Abstract

Facial performance capture is the process of automatically animating a digital face according to a captured performance of an actor. Recent developments in this area have focused on high-quality results using expensive head-scanning equipment and camera rigs. These methods produce impressive animations that accurately capture subtle details in an actor's performance. However, these methods are accessible only to content creators with relatively large budgets. Current methods using inexpensive recording equipment generally produce lower quality output that is unsuitable for many applications. In this paper, we present a facial performance capture method that does not require facial scans and instead animates an artist-created model using standard blendshapes. Furthermore, our method gives artists high-level control over animations through a workflow similar to existing commercial solutions. Given a recording, our approach matches keyframes of the video with corresponding expressions from an animated library of poses. A Gaussian process model then computes the full animation by interpolating from the set of matched keyframes. Our expression-matching method computes a low-dimensional latent code from an image that represents a facial expression while factoring out the facial identity. Images depicting similar facial expressions are identified by their proximity in the latent space. In our results, we demonstrate the fidelity of our expression-matching method. We also compare animations generated with our approach to animations generated with commercially available software.

CCS Concepts

• ***Computing methodologies*** → ***Animation;*** *Neural networks;*

1. Introduction

Facial performances are a key component of character animation where the expressiveness of the animation as well as the timing are both crucial for a sense of realism. Unfortunately, capturing high quality facial animations can be a time-consuming and expensive process. For animated film, the highest-quality solution typically involves animating a character's face by hand, giving the artist full control over the appearance and timing of a facial performance. Although every aspect of the performance is directable, a skilled artist might only produce a few seconds of animation per week.

For live action films, visual effects artists typically rely on highly detailed, actor-specific head scans acquired from sophisticated capture rigs consisting of large arrays of carefully calibrated cameras, such as the Light Stage [DHT*00] or the Medusa Facial Capture System [BHB*11]. Recently, data-driven approaches have been introduced that produce similar high-quality results for actors wearing head-mounted cameras [MHR17]. Additional work has allowed for facial performance capture to transfer an actor's performance onto a facial rig whose appearance does not match the actor's [HMB*18]. Although the resulting facial animations are often impressive, these methods are resource and time intensive, which limits their use to high-end productions such as feature films or high-budget video games. Lower budget productions have fewer options for generating facial animation, and there is often a noticeable difference in the animation quality.

Our work aims to improve the fidelity of facial performance capture methods while using inexpensive equipment. Additionally, we design our animation system to give artists control over the facial animation while maintaining a familiar context similar to existing facial performance capture tools. Our method does not require multi-camera facial capture systems or high-resolution scans. Instead, we rely on an artist-modeled blendshape facial rig, a set of manually posed expressions on the rig, and a set of recordings captured with an inexpensive helmet-mounted camera, thus bringing the cost to a fraction of that of high-end capture systems.

Our facial performance capture method operates in several stages. Our system first detects the face in a video frame, crops that part of the image, resizes it, and separates the face from the background. Next, a landmark detector processes the isolated face image to represent the expression as a sparse set of points. The landmarks are then passed to an expression-matching model which encodes these points into a low-dimensional latent space where facial expressions and identity are disentangled.

The user then selects keyframes from the recording. For each selected keyframe, our expression-matching model compares the facial landmark points to a set of landmark points detected on rendered images of the facial rig in latent space to find potential matching expressions with known rig controls that match the keyframes selected by the user. The user subsequently matches recorded expressions to rendered expressions and finally, the rig parameters from these matched expressions are interpolated across all frames of the recording to generate an animation.

Our contributions include a landmark detector designed specifically for encoding facial expressions, an expression-matching method that relies on these landmarks, and a user-guided method for generating facial animations given a set of matched expressions. For the landmark detector, we modify the first order motion model of Siarohin et al. [SLT*19] and use the landmark detector that is trained as a by-product. Our expression-matching model is implemented as an encoder-decoder neural network, and expressions are matched by computing distances from the model's latent codes.

We evaluate our approach on recordings of actors and artist-sculpted blendshape rigs and compare our results with other commercially available facial performance capture methods. We train the landmark detector with a set of facial recordings captured from a helmet-mounted camera and demonstrate that our method successfully generalizes to new actors and performances.

Our method uses a similar workflow to what artists are familiar with yet does not require the same level of technical skill to produce facial animations from a recording. With our system, a user picks keyframes from a recording and selects poses on a character rig that match the keyframes, similar to the process an artist would use with conventional animation software. In addition, the user can easily edit and refine an animation by adjusting selected keyframes and posed character expressions. The primary requirement of our system is an existing set of animation for the facial rig used for facial performance capture. An animator would need to create this data once for training our model. If animation data already exists for a character rig such as a pose library, which is commonly created as part of a character in animation studios, then that pose data could be used for training instead.

2. Related Work

A common approach for facial performance capture is first to fit a parametric model to an image of the actor's face and then to match the expression in the recorded face by finding the optimal rig parameters that minimize some objective function on the image. These approaches start with a parametric facial model, typically a morphable model [BV99, BRP*18], which solves the underconstrained problem of estimating facial geometry given a static image of a head. This type of parametric model can be created by scanning several hundred heads and then correlating their geometric differences through Principal Component Analysis (PCA). FaceWarehouse [CWZ*14] extends the morphable model into a multi-linear model by also including a statistical model of facial expressions. Once a 3D model has been fit to the actor's head, using for example the method of [MD19], the rig parameters are updated according to the displacement of image features from frame to frame by solving an optimization problem [LWP10, WBLP11, LYYB13, BWP13]. We refer the reader to the survey of Zollhöfer et al. [ZTG*18] for a thorough review of current approaches for monocular facial performance capture.

Recently, deep learning methods have helped improve results in facial reconstruction and performance capture. Deep video portraits [KGT*18] transfers the facial performance of an actor onto a target face by fitting a morphable model to the recordings and then applying deep learning models to synthesize a photorealistic image from a rasterized version of the facial model. Other methods [THMM17, TZK*17] train deep learning models to regress parameters of a morphable model given input images or video. In contrast, Laine et al . [LKA*17] use a neural network to output vertex positions of a facial model and avoid using an explicit morphable model. They collect data from high-quality facial scans for use during model training. Similarly, the codec avatar [LSSS18,WSS*19,SWW*20] generates facial geometry and view-dependent textures from a set of input videos. Unlike these previous approaches, our method instead generates an animation by matching expressions of a recorded actor with corresponding pre-made expressions on a facial rig in image space.

Style transfer is a key component of our method and can be

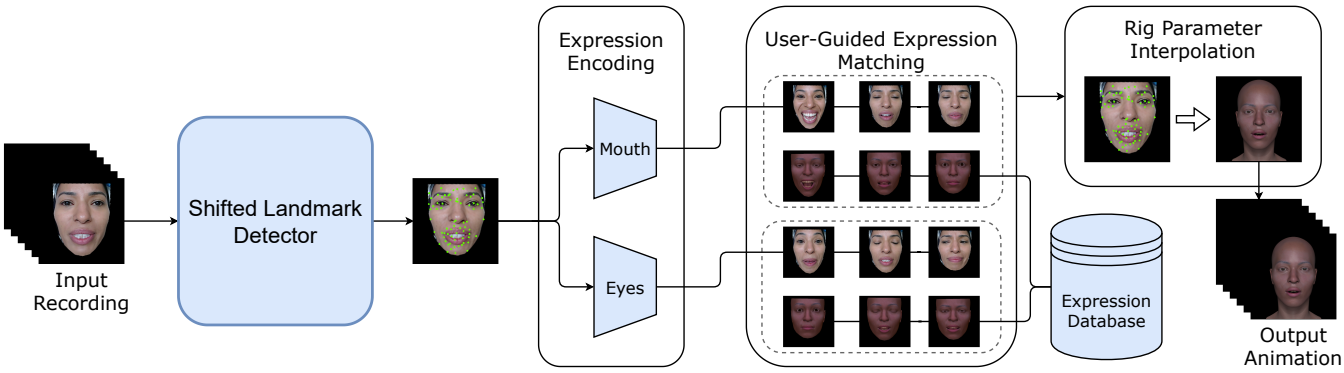


Figure 2: *Visualization of our facial performance capture pipeline. First, the input passes through a shifted landmark detector (Sec. 5.4) that outputs landmark points that are statistically similar to the landmarks from our training dataset. Next, the expression encoder computes latent codes from these computed landmark points. Afterwards, keyframes and matching expressions are identified through a user-guided process. Finally our system generates an animation by interpolating between the keyframes.*

loosely defined as rendering the content of one image in the style of another. Gatys et al . [GEB16] introduced an iterative method for style transfer by optimizing correlations between features from intermediate layers in pretrained Convolutional Neural Networks (CNN). Subsequent work replaced the iterative optimization step with deep learning models to allow for real-time style transfer [JAFF16, ULVL16, UVL16]. Other work includes pix2pix [IZZE17], which proposes a solution to the image-to-image translation problem with paired training data. CycleGAN [ZPIE17] addresses the same image-to-image translation problem but works in an unsupervised setting. StarGAN [CCK*17] addresses the multi-domain problem by utilizing pre-defined labels for images across many different style domains. More recently, StarGAN v2 [CUYH20] replaces the domain labels from StarGAN with style codes that are learned during model training. In the context of digital humans, StyleGAN [KLA19] can synthesize realistic facial images from randomly generated latent vectors. However, because the model is a GAN, users typically have no direct control over the images generated by the model.

Additional research has addressed style transfer specifically for digital portraits of humans. These approaches generate facial images through image processing alone and do not rely on geometric priors such as a morphable model. X2Face [WKZ18] develops an encoder-decoder network to learn a latent code for facial images. Similarly, DR-GAN [TYL17] learns latent codes that separate facial variations such as expression from a person's identity. Other work [ZSBL19] synthesizes novel facial images given a source photo and the landmark positions of the target pose. CariosGAN [CLY18] synthesizes caricature images through a style transfer component and a geometric component, which warps the style-transferred image based on exaggerated movement of facial landmarks.

Closer to our work, Moser et al . [MCW*21] recently proposed a two-step process for facial performance capture. The first step utilizes style transfer to match a recorded actor and rendered model in image space, and the second step regresses rig parameters from the style transferred image. Similarly, our method relies on style transfer to copy facial expressions across different identities. However, we use style transfer to generate training data and instead rely on facial landmarks to match expressions for animation synthesis.

For the style transfer component of our method, we use the first order motion model of Siarohin et al. [SLT*19]. This approach detects motion between pairs of images through an unsupervised landmark detector. An image generator then uses the motion to warp image features in order to synthesize a new image of some target style with the driving motion from an image pair. We found this approach works well with our method because it preserves the appearance of the facial geometry during style transfer.

Another key component of our method is facial expression matching. Recently, methods have been developed that use deep learning techniques for facial expression recognition [LD20]. These types of approaches use a dataset of images labeled with one of several basic emotional expressions. They then train a classifier model to identify the expression depicted in input images. Instead of building a classifier model, Vemulapalli and Agarwala [VA19] propose a method to embed images in a low-dimensional space based on facial expression similarity. Our expression matching scheme is most similar to this approach but does not require a manually labeled dataset of emotional expressions depicted in photographs.

3. Overview

Our facial performance capture method consists of two primary components: a style transfer model along with a landmark detector and a facial expression matching model. The style transfer model generates training data for the expression matching model by synthesizing images of different facial identities with similar expressions. The landmark detector is a byproduct of the style transfer model which encodes images as a sparse set of points used as inputs to the expression matching model. We implement this expression matching model as an encoder, which represents expressions as latent codes and maps similar facial expressions to the same region in the latent space.

After a video has been processed into latent codes through the expression encoder, our facial performance capture method generates an animation through a user-guided process. During this process, the user is asked to select keyframes throughout a recorded video. For each keyframe (typically 1 per second of footage in our experiments), the user selects the best-matching facial expression from a small number of poses proposed by the expression matching model. The proposed expressions correspond to artist-created poses on a character rig. We finally interpolate poses between the keyframes on the facial rig to generate the output animation. Figure 2 shows the pipeline of our method.

4. Style Transfer

4.1. Style Transfer Model

We derive our style transfer approach from the first order motion model [SLT*19], which transfers motion from a video sequence onto a separate subject. The model consists of three components: an unsupervised landmark detector, an optical flow field generator, and an image generator. Our primary changes to this model are in the loss function used for training, with a focus on the landmark detector component.

We optimize the model parameters using the perceptual loss as well as the equivariance constraint proposed by the authors of the first order model. Additionally, we introduce a landmark distance loss (L_{dist}), a background-deterring loss (L_{bg}), and a regularization term (L_{reg}) to promote a better distribution of landmarks across the face. Figure 3 shows a comparison of landmarks points detected by models trained with and without these additional loss terms. During training, the model learns to reconstruct the same images provided as input. Given a driving image $\mathcal{V}_d$ and the reconstructed image $\hat{\mathcal{V}}_d$ generated by the first order model, the full loss function is

$$\begin{aligned}\mathcal{L}(\mathcal{V}_d,\hat{\mathcal{V}}_d) =& L_{percep}(\mathcal{V}_d,\hat{\mathcal{V}}_d)+L_{equi}(\mathcal{V}_d)+\\ & L_{dist}(\mathcal{V}_d)+L_{bg}(\mathcal{V}_d)+L_{reg}(\mathcal{V}_d).\end{aligned} \tag{1}$$

For the landmark detector, to generate k landmark coordinates, the model first produces one heatmap for each landmark. Next, a softmax operation limits the range of heatmap values to produce a set of confidence maps $C(\mathcal{V}_d) \in [0,1]^{h\times w\times k}$. Because our use-case only requires that the face in an image be accurately reconstructed, the landmark detector should place all points on the face and ignore the background in an image. To achieve this goal, we first replace the background in the input image with solid black to remove any unnecessary image features. Then, we apply a background-deterring constraint to the confidence maps as follows:

$$L_{bg}(\mathcal{V}_d) = \frac{\lambda_{bg}}{khw}\sum_{i=1}^{k}\sum_{y=1}^{h}\sum_{x=1}^{w} C_i(\mathcal{V}_d)[x,y]\cdot\mathcal{M}(\mathcal{V}_d)[x,y] \tag{2}$$

where $\lambda_{bg} > 0$ is a user-defined hyperparameter. $C_i(\mathcal{V}_d)[x,y]$ indicates the pixel value at the coordinate (x,y) in the confidence map for landmark i, and $\mathcal{M}(\mathcal{V}_d) \in \{0,1\}^{h\times w}$ is a binary mask that is nonzero for pixels lying on the background of the image. Figure 4 shows an example of a video frame with the background removed along with the mask $\mathcal{M}$.

The distance error $L_{dist}(\mathcal{V}_d)$ penalizes overlapping landmarks and landmarks that are in close proximity. Given the set of k landmarks $\boldsymbol{\mu}(\mathcal{V}_d)$ estimated by the landmark detector, we compute the pair-wise squared euclidean distance between the landmarks $d_{ij} = \left\|\boldsymbol{\mu}_i(\mathcal{V}_d) - \boldsymbol{\mu}_j(\mathcal{V}_d)\right\|_2^2$. Thus, the distance loss is

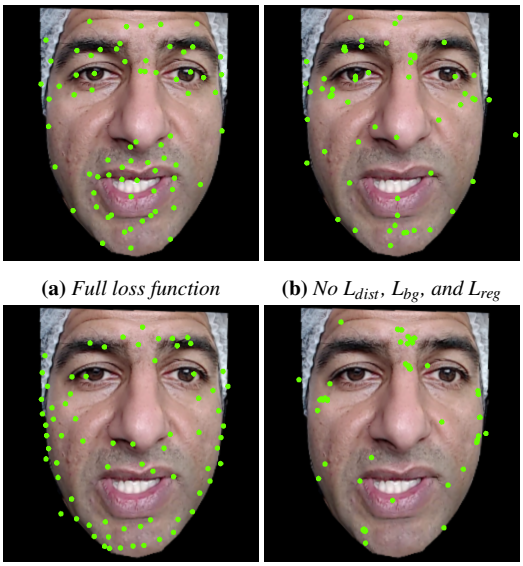
$$L_{dist}(\mathcal{V}_d) = \frac{\lambda_{dist}}{k(k-1)/2}\sum_{i=1}^{k}\sum_{j=i+1}^{k}\exp(-d_{ij}T) \tag{3}$$

where $\lambda_{dist} > 0$ and $T > 0$ are user-defined hyperparameters.

In addition to landmarks $\boldsymbol{\mu}(\mathcal{V}_d)$, the model also outputs k 2×2 transformation matrices $\mathbf{J}(\mathcal{V}_d)$ that estimates the first order deformations at each landmark point in the image. Our regularization term $L_{reg}(\mathcal{V}_d)$ penalizes large eigenvalues in the transformation matrices $\mathbf{J}(\mathcal{V}_d)$. Large eigenvalues indicate substantial scaling along the corresponding eigenvectors which creates undesirable stretching distortions. On human faces, the mouth region typically exhibits the largest deformations across various poses, and the transformation matrices can reflect information about stretched and deformed parts of the mouth. However, we found that penalizing large transformations in $\mathbf{J}(\mathcal{V}_d)$ produces visually better results in our method. To regularize these matrices, we apply the following loss:

$$L_{reg}(\mathcal{V}_d) = \frac{\lambda_{reg}}{k}\sum_{i=1}^{k}\|\mathbf{J}_i(\mathcal{V}_d) - I\|_F^2 \tag{4}$$

where I is the 2×2 identity matrix. Although we could penalize large eigenvalues in $\mathbf{J}(\mathcal{V}_d)$ with a rotation-invariant loss, we found that Equation 4 is simple to implement, produces good results, and can easily be differentiated.

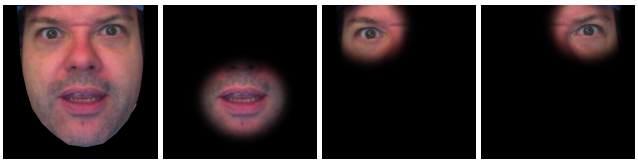


(a) *Full loss function* **(b)** *No L_{dist}, L_{bg}, and L_{reg}*

(c) *No segmentation* **(d)** *Original First Order*

Figure 3: *Visualization of landmark points generated by a model trained on the full objective function (a), a model trained without the distance, background, and regularization loss (b), a model trained without facial segmentation (c), and the original first order motion model (d).*

Figure 4: *Example of a recorded frame (left) with the background removed (middle) and the mask $\mathcal{M}$ (right) used for the background-deterring objective.*



(a) **(b)** **(c)** **(d)**

Figure 5: *Example of an input frame (a) with a mask applied to the mouth (b), the right eye (c), and the left eye (d).*

4.2. Facial Segmentation

Although the style transfer network can operate on the full image of a face, we found segmenting the face into regions and training a separate model on each facial region produces more plausible animations. For segmentation, we train a model for the full face, a model for the mouth, one model for the left eye and eyebrow, and one for the right eye and eyebrow. A separate off the shelf facial landmark detector processes input images to identify these facial regions. Figure 5 shows an example frame along with masks applied to the three separate regions of the face. Our idea of training separate models on semantic facial regions is similar in spirit to the approach of [CBGB20] who show the benefits of training per-region networks for landmark detection, especially for high resolution images. Intuitively, facial segmentation is how an artists would operate on the input image if they were tasked with manually transferring the expressions from source to target, by iteratively zooming in and out of the region of interest they are currently working on.

For each region in an image, we identify landmarks and compute the convex hull containing those points. The convex hull is then expanded by a user-defined margin, and the shape masks the image so that only the relevant facial region is visible. During model training, the input is still the unmasked images. However, the objective function (Equation 1) operates on masked versions of the driving image $\mathcal{V}_d$ and the reconstructed image $\hat{\mathcal{V}}_d$. Thus, the model has access to the full image but only needs to output the image within the masked region. Figure 3 shows a comparison of detected landmark positions for models trained with (a,b) and without (c,d) facial segmentation. For the style transfer models, we use 24 landmark points for the models trained on the full face and mouth and 12 points for the models trained on each eye.

When evaluating the model for style transfer, we merge the results together to construct a single image. First, the model of the full face processes the input. Next, the remaining style transfer models process the image separately. Our system then merges the outputs according to the masks of the input driving image. Pixels are blended with the combined image according to the regional masks' weights.

4.3. Model Training

We train the model on a set of rendered images featuring a diverse set of facial expressions and identities as well as a set of images of actors covering many identities and expressions. The rendered images are rasterized from blendshape models posed with randomly generated expressions. The camera parameters and lighting conditions are constant across all images. The random expressions are generated by augmenting a relatively small set of 104 artist-created poses. To synthesize a new pose, we first divide the blendweights into two sets: one set for the eyes and upper half of the face and another set for the mouth and the lower half of the face. Next, we select two poses uniformly at random from the set of artist created poses. For each pose with probability 0.5, we randomly choose to mirror the expression by flipping the left and right rig controls. A new pose is constructed by combining the set of upper half blendweights from one pose with the set of lower half blendweights from the other pose. Finally, we add uniformly random noise to all non-zero blendweights in the synthesized pose. The noise is drawn i.i.d. from the distribution $U(-0.15, 0.15)$, and the resulting blendweight values are clamped to the range $[0, 1]$.

All input images are resized to a resolution of 256x256 and we start by detecting landmarks with a model trained on a distribution of 68 facial landmarks as defined in the iBUG 300-W dataset [SAT*16]. Next, we crop each image so that the image is square and so that the face occupies the center of the image. We then compute the convex hull of the landmarks and set to black any pixel that lies outside of it. During training of the style transfer model for each sample, we randomly pick with 0.5 probability to use either one of the rendered image sets or one of the recorded image sets.

For the training hyperparameters, we use the same values as those used by the authors of the first order motion model. We additionally set $\lambda_{bg} = 5$, $\lambda_{dist} = 5$, $\lambda_{reg} = 5$, and $T = 300$.

5. Facial Animation

We use a kernel-based method to animate a facial rig from a recorded performance. Using a set of artist-posed expressions, we identify matching facial expressions between a keyframes from the recorded actor and renderings of the artist-created facial expressions. Our method then uses a Gaussian process model to interpolate the rig parameters for the remaining frames of the recording.

5.1. Expression Matching

To match expressions, we reduce images to low-dimensional latent codes and measure similarity between images through Euclidean distances in this latent space. We compute latent codes through a two-step process. The style transfer model first computes landmarks positions and transformation matrices from an input image.

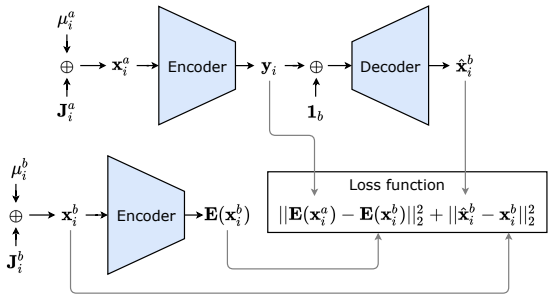


Figure 6: *Diagram of the expression encoder-decoder model during training. The inputs to the model $\boldsymbol{\mu}_i^a$ and $\mathbf{J}_i^a$ are the landmark positions and transformations computed from the image of expression i on facial identity a. These inputs are flattened and concatenated ($\oplus$) to form the vector input $\mathbf{x}_i^a$ to the encoder. The encoder generates the latent code $\mathbf{y}_i$. This vector is concatenated with a one-hot vector encoding $\mathbf{1}_b$ to provide the decoder with facial identity information. Finally, the output $\hat{\mathbf{x}}_i^b$ is the model's estimation for the landmark positions and transformations for expression i and identity b. During training, the model also encodes the landmarks and transformations of expression i and identity b in order to minimize the difference between the latent codes.*

Second, an encoder generates a latent code from the landmark positions and transformations. We implement the encoder as a neural network and train it as part of an autoencoder model. The model learns to encode facial expressions while maintaining invariance to changes in facial identity. For example, the encoder maps images of two visually different actors performing a smile to the same latent code. Figure 6 illustrates the encoder-decoder architecture.

The training dataset consists of images of m different facial identities, each posed for an identical set of n facial expressions. Thus, the dataset includes $m \times n$ images where image $\mathcal{V}_i^j$ corresponds with facial expression i and identity j. Given $\mathcal{V}_i^j$, the first order motion model computes landmark positions $\boldsymbol{\mu}(\mathcal{V}_i^j)$ and transformation matrices $\mathbf{J}(\mathcal{V}_i^j)$. These positions and matrices are then flattened and concatenated to form the input $\mathbf{x}_i^j$ to the encoder which then generates a low-dimensional latent code $\mathbf{y}$. The decoder's input consists of $\mathbf{y}$ concatenated with the one-hot encoding of the facial identity $\mathbf{y}^j = [\mathbf{y}, \mathbf{1}_j]$. By supplying the decoder with the facial identity, the encoder does not need to express this information in the latent code.

The encoder-decoder model learns to reconstruct landmark positions and transformation matrices of a specific identity when given the landmarks of the same expression but a different identity. Let the output of the model be $\hat{\mathbf{x}}_i^b = \mathbf{D}(\mathbf{E}(\mathbf{x}_i^a), \mathbf{1}_b)$ where i is the facial expression, a and b are the identities of the input and the output, respectively, $\mathbf{E}$ is the encoder model, and $\mathbf{D}$ is the decoder model. Model training consists of a loss function that penalizes differences in the reconstructed vector as well as differences in encoded latent vectors across different identities:

$$L(\mathbf{x}_i^a, \mathbf{x}_i^b) = \left\| \mathbf{E}(\mathbf{x}_i^a) - \mathbf{E}(\mathbf{x}_i^b) \right\|_2^2 + \left\| \hat{\mathbf{x}}_i^b - \mathbf{x}_i^b \right\|_2^2. \tag{5}$$

Figure 6 shows a graphical representation of the loss function.

Once trained, we use the encoder to identify similar facial expressions by comparing Euclidean distances in the latent space. Specifically, given an image $\mathcal{V}$ and the detected landmark positions and matrices, the closest expression i for identity j in the dataset is

$$i = \underset{i \in \{1,2,\ldots,n\}}{\arg\min} \left\| \mathbf{E}(\mathbf{x}) - \mathbf{E}(\mathbf{x}_i^j) \right\|_2 \tag{6}$$

where $\mathbf{x}$ is the concatenation of the flattened landmarks $\boldsymbol{\mu}(\mathcal{V})$ and matrices $\mathbf{J}(\mathcal{V})$. To simplify the expression-matching problem, we split the encoder model segments: one for the eyes and upper half of the face and one for the mouth and lower half and train two separate models, one for each half of the face.

5.2. Expression Interpolation

With a set of keyframes and matched expressions for these frames, we propose using a Gaussian process model to interpolate rig parameters between keyframes to animate the facial rig according to a recorded performance. We define a facial rig as a function $\mathbf{M} = \mathbf{r}(\mathbf{p})$ that outputs a mesh $\mathbf{M}$ deformed according to underlying rig parameters $\mathbf{p}$. With a set of n posed facial expressions $[\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n]$, we render the deformed meshes to generate a set of images $[\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n]$. Let a recorded sequence of m frames of an actor's facial performance be represented as $[\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_m]$, and let $[(\mathcal{V}_1', \mathcal{I}_1'), (\mathcal{V}_2', \mathcal{I}_2'), ..., (\mathcal{V}_q', \mathcal{I}_q')]$ be a set of q paired images between the rendered and recorded sets in which the facial expressions match. For each paired match $(\mathcal{V}_i', \mathcal{I}_i')$, we compute the landmark positions and transformations on the recorded performance frame $\mathcal{V}_i'$ to construct the encoder input and latent code $\mathbf{y}_i' = \mathbf{E}(\mathbf{x}_i')$. We then construct a set of latent codes paired with corresponding rig parameters: $[(\mathbf{y}_1', \mathbf{p}_1'), (\mathbf{y}_2', \mathbf{p}_2'), ..., (\mathbf{y}_q', \mathbf{p}_q')]$.

With this set of paired data, we then construct a Gaussian process model to predict rig parameters from latent codes. We construct the kernel matrix $\mathbf{K} \in \mathbb{R}^{q \times q}$ using the radial basis function such that

$$k_{ij} = \exp\left(\frac{-\left\| \mathbf{y}_i' - \mathbf{y}_j' \right\|_2^2}{2\sigma^2} \right) \tag{7}$$

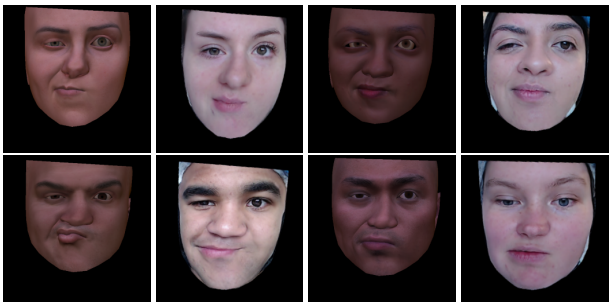
where $\sigma \in \mathbb{R}$ is a user-defined width. Given a new frame $\mathcal{V}^*$ from the recorded performance, we first compute the latent code $\mathbf{y}^*$ from the encoder and estimate the rig parameters $\mathbf{p}^*$ as the mean of the Gaussian process model evaluated at $\mathbf{y}^*$

$$\mathbf{p}^* = \mathbf{k}^* \mathbf{K}^{-1} \mathbf{Y} \tag{8}$$

where $\mathbf{Y}$ is a stacked matrix of rig parameters from the paired set and $\mathbf{k}^* \in \mathbb{R}^q$ is a vector where $k_i^* = \exp(-(2\sigma)^{-2} \left\| \mathbf{y}^* - \mathbf{y}_i' \right\|_2^2)$.

5.3. Model Training

The accuracy of our expression matching method depends on the quality of the dataset used to train the encoder-decoder model. To ensure that expressions remain identical across all facial identities, we construct the dataset from rendered images. First, we collect a large set of expressions posed on a single facial rig. Next, we build a set of facial rigs that differ in visual identity but all share the same underlying controls. The set of expressions are then directly transferred to all of the facial rigs through the rig parameters. Thus, the



(a) (b) (c) (d)

Figure 7: *Examples of an expression transferred from a rendered blendshape model (a) and (c) to a recorded actor (b) and (d), respectively, through the first order motion model.*

expressions across different facial identities closely match. We augment the dataset with expressions transferred onto recorded images through the first order motion model. Asking an actor to match an expression exactly is a difficult task so we rely on style transfer instead to generate realistic-looking images of specific expressions. Figure 7 shows example images generated through style transfer. In total, our dataset consists of 14 different facial rigs and images of expressions transferred onto 14 different recorded actors.

We train the encoder-decoder model using the Adam optimizer [KB14]. The model is trained for 30,000, 15,000, and 15,000 iterations with learning rates of 2×10^{-3}, 5×10^{-4}, and 5×10^{-5}, respectively. We use a batch size of 8, and each sample consists of two vectors: the encoder input $\mathbf{x}_i^a$ and the decoder's target output $\mathbf{x}_i^b$. We additionally perturb the input vector $\mathbf{x}_i^a$ to help the model generalize to new inputs. We apply random rotation, scaling, and translation to the input's landmark positions and transformation matrices. The rotation angle is drawn uniformly at random from $[-10°, 10°]$. The scale factor is drawn uniformly at random from $[0.925, 1.075]$. The X and Y components of the translation are drawn uniformly at random from $[-0.025w, 0.025w]$ where w is the width of the image.

The encoder consists of three dense hidden layers with the ReLU activation function applied between each layer. The three layers consist of 64, 48, and 32 hidden units, respectively. A final dense layer is applied to produce the latent code in a 32 dimensional space. The decoder mirrors the encoder with three layers, each consisting of 32, 48, and 64 hidden units, respectively. A final dense layer is applied to transform the code into the original input space.

5.4. Domain Shift

The training set for the expression matching model consists of landmarks computed from images output by the first order motion model. However, during evaluation, the model receives landmarks computed directly from recorded video frames. As a result, the expression matching model performs poorly when evaluated on landmarks computed directly from a recorded video due to domain shift. To alleviate this problem, we propose a "shifted landmark detector" in which the first order motion model transfers the style of the recording onto the same recording. The result is a video almost identical to the original. The landmark detector then processes these new frames. The new landmarks are now statistically more similar to the training data than landmarks computed from the original recording.

6. Results and Evaluation

To evaluate our method, we first construct a dataset of video recordings for training the style transfer model and the expression matching model. For style transfer, our training set of video recordings consists of 453 minutes of facial performances across 59 actors. The videos are captured with an inexpensive webcam and are recorded at a rate of 60 FPS with a resolution of 1280×720. Most of our results were generated with the camera mounted on a helmet worn by the actor to eliminate head rotation relative to the camera. However, we show in the supplemental video an animation generated from footage of an actor sitting in front of a webcam mounted on a screen in front of them. Our system works as long as the actor's face does not rotate too far away from the camera's optical axis.

Our facial rigs are linear blendshape models. Each blendshape model consists of a set of meshes $\mathbf{B} = [\mathbf{b}^0, \mathbf{b}^1, ...\mathbf{b}^m]$, where the mesh $\mathbf{b}^0$ is a neutral expression and mesh $\mathbf{b}^i$, $i > 0$, is some artist-defined expression such as "open mouth". The mesh deformation can be parameterized by a vector of rig parameters, called blendweights, $\mathbf{p} \in \mathbb{R}^m$ and is computed as

$$\mathbf{r}(\mathbf{p}) = \mathbf{b}^0 + \sum_{i=1}^{m} \left(\mathbf{b}^i - \mathbf{b}^0\right) p_i. \tag{9}$$

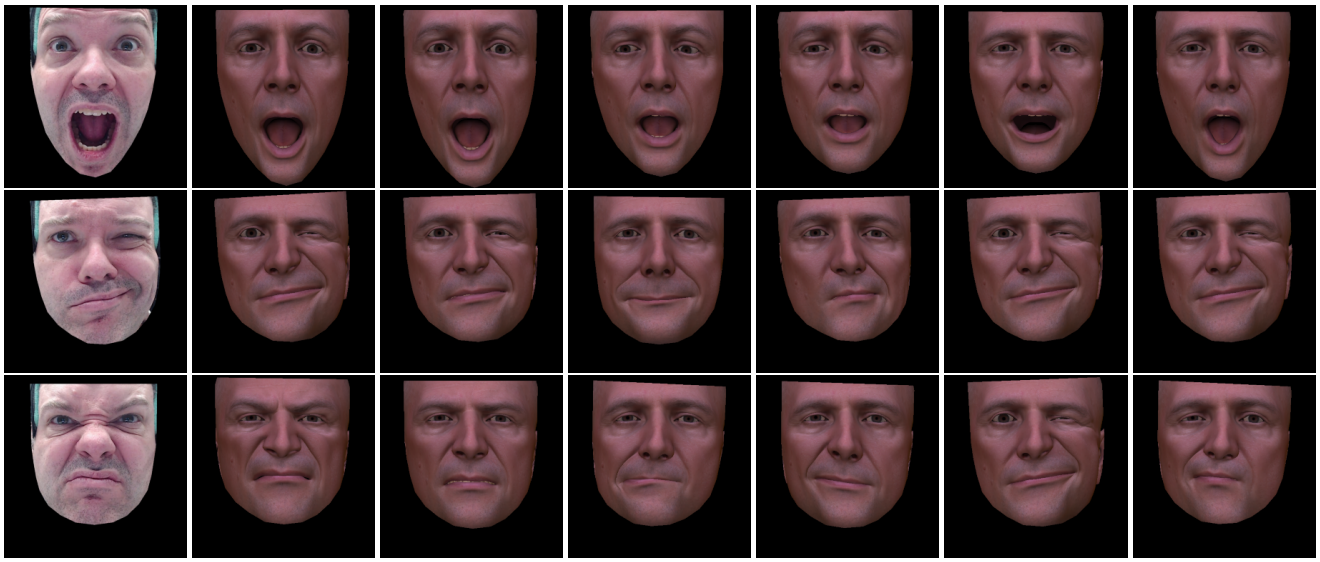
Our artist-created facial blendshape rigs are modeled after 14 of the actors in the training recordings. The models use identical mesh topology and consist of 11,406 vertices, and each rig utilizes 92 blendshapes. To generate the training images for the style transfer model, an artist manually creates 104 poses that cover a wide range of emotional expressions and the full range of visemes of the blendshape model. We augment the poses using the method described in Section 4.3 to produce a set of 2,000 samples.

Our method is implemented in Python with TensorFlow for the deep learning components. We run our method on a 20 core machine running at 2.20 GHz with one Nvidia Titan RTX GPU. Training the four style transfer networks takes roughly one day in total. Training the expression matching model takes 10-20 minutes.

6.1. Expression Matching Accuracy

The quality of the resulting animation depends on how well the model can map images of matching expressions to similar latent codes. Our expression matching model depends on 2D landmark positions and transformation matrices that are computed from images. We compare different types of inputs and demonstrate that landmarks generated from our style transfer model provide the best inputs for the expression matching model in our experiment.

We recorded an actor's face for a 5 minute range of motion performance to train the expression matching encoder-decoder model. An artist animated the performance, and the parameters were transferred to all of the rigs. The style transfer model was used to apply the animation to recorded actors for additional training examples.



(a) *Recorded* **(b)** *Animated* **(c)** *Ours* **(d)** *No Seg* **(e)** *No Extra Loss* **(f)** *First Order* **(g)** *Landmarks*

Figure 8: *Examples of recorded expressions with corresponding artist-created poses (Animated) and matched poses from expression matching models evaluated on landmarks from our style transfer model (Ours), our style transfer model trained without segmentation (No Seg), the first order motion model trained with segmentation (No Extra Loss), the original first order motion model (First Order), and a landmark detector trained in a supervised setting (Landmarks).*

We evaluate expression matching models trained on different landmark encodings. We train five separate models on landmark data generated from our style transfer model, the original first order motion model, two other variants, and a separate state-of-the-art facial landmark detector [FKA*18]. The first of the two variants is trained with a landmark detector optimized with our modified loss function from Equation 1 and without facial segmentation as described in Section 4.2. The second variant is trained with a landmark detector optimized with segmentation and with the original objective function from the first order motion model.

For our evaluation, let $\{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_n\}$ represent a recorded performance of length n frames, and let $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n\}$ represent the rendered images of the corresponding artist-created animation. Thus, images $\mathcal{V}_i$ and $\mathcal{I}_i$ share the same facial expression.

The encoder-decoder model's accuracy is evaluated by the proximity of the latent codes $\mathbf{E}(\mathbf{x}_i)$ and $\mathbf{E}(\mathbf{x}_i{}')$ for frame i. Because the latent spaces between different encoders are not guaranteed to be similar, we cannot directly compare the models through distances between latent codes. Instead, we evaluate accuracy as the distance between matching expressions relative to all other expressions in the recording. Let $\{\mathbf{E}(\mathbf{x}_1), \mathbf{E}(\mathbf{x}_2), ..., \mathbf{E}(\mathbf{x}_n)\}$ represent the set of latent codes evaluated on the recorded performance, and let $\{\mathbf{E}(\mathbf{x}_1{}'), \mathbf{E}(\mathbf{x}_2{}'), ..., \mathbf{E}(\mathbf{x}_n{}')\}$ represent the set of latent codes evaluated on the rendered animation. For frame i, we compute the distance between the latent code of the recorded frame with the codes of all rendered frames:

$$d_{ij} = \left\| \mathbf{E}(\mathbf{x}_i) - \mathbf{E}(\mathbf{x}_j{}') \right\|_2 . \tag{10}$$

Next, we sort the set $\{d_{i1}, d_{i2}, ..., d_{in}\}$ in ascending order where $c_i(k)$ indicates the position of frame k in the sorted set of distances evaluated on frame i. We evaluate a model's accuracy through the position $c_i(i)$ of the corresponding rendered expression i in the sorted list. For each model, we evaluate the accuracy as the average index position of matched expressions:

$$\frac{1}{n^2} \sum_{i=1}^{n} c_i(i) \tag{11}$$

where a lower value is better. For this accuracy measure, we use encoders trained on full facial expressions, instead of one for the lower half and one for the upper half of the face.

Table 1: *Expression matching accuracy measured on the recorded range of motion performance, the animation style transferred onto a recorded actor, and a performance of an actor talking. The accuracy is evaluated according to Equation 11 on models trained on inputs from five different landmark detectors: our style transfer model (Ours), our style transfer model trained without segmentation (No Seg), the first order motion model trained with segmentation (No Extra Loss), the original first order motion model (First Order), and a landmark detector trained in a supervised setting (Landmarks).*

	Range of Motion	Style Transfer	Talking
Ours	**0.11**	**0.10**	**0.09**
No Seg	0.19	0.12	0.18
No Extra Loss	0.19	0.20	0.19
First Order	0.20	0.20	0.27
Landmarks	0.19	0.13	0.16

(a) *Recorded* **(b)** *Ours* **(c)** *No Seg* **(d)** *No Extra Loss* **(e)** *First Order* **(f)** *Landmarks*

Figure 9: *Examples of recorded expressions compared with matched poses from expression matching models evaluated on landmarks from our style transfer model (Ours), our style transfer model trained without segmentation (No Seg), the first order motion model trained with segmentation (No Extra Loss), the original first order motion model (First Order), and a landmark detector trained in a supervised setting (Landmarks).*

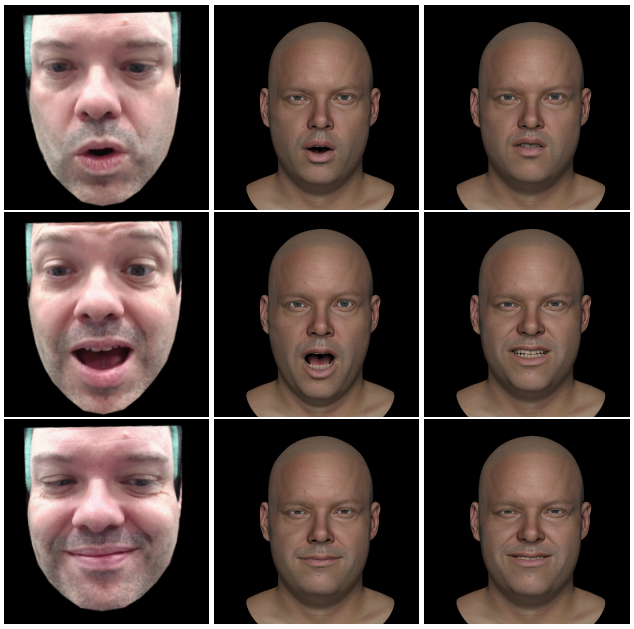
We evaluate the expression matching models on three recordings. The first is the original range of motion performance upon which the training animation is based. The second recording is synthesized by transferring the style of a rendered range of motion animation onto a recorded actor. Images of the facial rig and of the recorded actor used for this synthesized video were excluded from the training set. The third recording is a separate video of a different actor talking with varying facial expressions. Table 1 shows the accuracy of the encoder-decoder model trained on five different sets of input landmarks. In each test, our method produces the most accurate results. Figure 8 compares several recorded expressions with the corresponding frame of the artist-created animation and the closest matching expression computed from each expression matching model. Figure 9 shows additional matched expressions from other recordings.

6.2. Evaluation Against Commercial Applications

We develop a user-guided animation system to compare our method against two commercially available animation applications: ARKit and Dynamixyz. ARKit animates blendshape models with 52 predefined facial expressions in real-time on a mobile device. Unlike our method however, it requires depth information which requires more sophisticated hardware than a simple RGB feed from a camera.

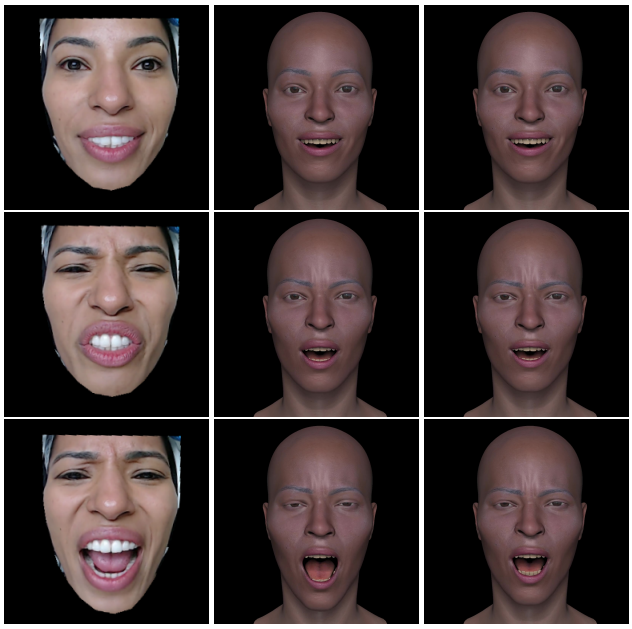
On the other hand, Dynamixyz animates a facial rig through a user-guided process. For a recorded performance, the software detects facial landmarks and allows the user to correct any inaccurately placed landmarks. Next the user picks keyframes from the recording and poses the blendshape model to match the expressions in the keyframes. Finally, Dynamixyz generates an animation by interpolating the keyframes according to the landmark positions in each frame.

To generate an animation with our method, we develop a tool similar to Dynamixyz. We use facial landmarks from the style transfer model, which removes the requirement for user interaction for landmark placement. For a recorded performance, the tool collects user-selected keyframes and Instead of requiring the user to pose a facial rig according to the keyframes, our tool relies on ex-



(a) *Driving video frame* **(b)** *Ours* **(c)** *ARKit*

Figure 10: *A visual comparison of our method with ARKit, a commerical software for low-cost performance capture. As can be seen, our method surpasses ARKit in terms of visual quality and faithfulness to the driving frames.*



(a) *Driving video frame* **(b)** *Ours* **(c)** *Dynamixyz*

Figure 11: *A visual comparison of our method with Dynamixyz. We show that our method can produce facial expressions similar to what a skilled artist would create using professional animation software.*

pression matching to suggest existing posed expressions on the rig. The proposed rig poses are selected from an artist-generated animation covering a wide range of facial expressions. With the set of keyframes and matched poses, the process described in Section 5.2 generates the full animation. To work with the expression encoding models trained on the upper half and lower half of the face, we divide the rig parameters according to these facial regions and animate each half separately. In our animation tool, the user separately selects keyframes and matched poses for the two facial regions.

When evaluating the expression matching model, the numerically best estimated pose might not be the best visual match. To account for this potential problem, our tool presents the user with a set of different poses, and the user selects the pose that best matches a keyframe visually. When presenting the user with potential pose matches, the tool picks a small subset from the artist-created animation database. The tool selects poses by picking ones that have similar latent codes at different times in the animation database. In our implementation, we experimentally chose to present the user with six potential matches for each keyframe to guarantee a good trade-off between faithfulness and time and effort required to generate the final animation.

Figure 10 shows a qualitative comparison of our method against ARKit, where the actor drives an avatar of his own likeness. As can be seen, our method provides more faithful animations when compared to ARKit. Visually, ARKit either under (second row) or over evaluates mouth expressions (third row) as well as sometimes outputs the wrong expression. This is the case in the first row of Figure 10 where the lip funnel, which our method properly captures, is turned into a gnarl by ARKit. Furthermore, the accompanying video shows that our method generates an animation that is more natural and faithful to the driving video.

To compare our method with Dynamixyz, we supplied an artist with a 54 second performance and a blendshape rig matching the appearance of the recorded actor. The artist has nearly 20 years of experience working as a professional animator. He spent 2 to 3 weeks to create the animation through Dynamixyz. Thus, this animation represents high-quality, professional work. In contrast, the animation generated by our method was generated with roughly one hour of the user's time, and the user had no formal training in animation. Figure 11 shows three frames from the recording for comparison. Please see the accompanying video for the full comparison. Although the animation produced through Dynamixyz looks visually better, our method shows promising results when considering that our animation required less time to author and could be accomplished by someone not trained in the art of character animation.

7. Discussion

We have presented a method for facial performance capture consisting of a style transfer component and an expression-matching component. Our method does not require high-quality facial scans nor does it require that a recorded actor's appearance match that of the target facial rig. Instead, our method works with blendshape models, example poses covering a wide range of expression on the rig,

and a training set of facial performances recorded from a helmet-mounted camera. As a result, actors of various appearances can drive a facial rig through our performance capture method without retraining models specific to each actor's appearance.

Our method closely follows the workflow of existing tools so that artists can quickly learn how to use our animation system. Furthermore, our method automates some of the time consuming aspects of other facial performance capture software so that artists can author animations significantly faster. Because our method does not require the user to build a pose library specific to each actor and recording, the user does not need to create poses on the facial rig during the animation process. Thus, users without prior animation experience can use our tool. A skilled animator would only be needed to create the initial training data.

Currently, our facial performance capture method requires some user interaction. We require user input because our expression-matching model occasionally matches images that do not depict similar expressions. Although this type of failure is not a common occurrence, a single mismatched expression could result in undesirable artifacts in the animation. If the expression-matching model's accuracy is further improved, then our method could run fully automatically.

Because our expression matching model compares single frames in isolation, the best matches that it finds could lose important contextual information in a performance. For example, in a performance of an actor talking, an animator can identify phonemes through the audio and activate corresponding shapes to produce the desired mouth appearance. Our method, in contrast, could find visually similar mouth appearances that do not match the correct visemes in the performance. Furthermore, prior research has established that the mouth region is problematic even for more sophisticated facial reconstruction methods in controlled environment such as [BHB*11], where improvements later came at the added cost of lip-specific, user-guided correctives [DBB*18]. Given that our method aims to provide for general facial animation, we leave it as future work to incorporate additional information, such as audio or temporal coherency, to help produce animations more similar to what an artist would create.

Although our method produces believable animations from recordings, our approach cannot animate eyeball rotations from a performance. The main limitation for this lack of eye tracking lies in the style transfer model. We found that in some cases, eyeball rotations are not accurately transferred between images. As a result, the expression-matching model is unable to represent eye rotations in the latent code due to the noise in the input data. Future improvements could explore work to allow style transfer of eyes. In the supplementary video, our results show eye animation derived from a separate process for illustration purposes only. We use an off the shelf eye tracking solution to locate pupil centers in a recording and map the pupil centers to eye rotations on the character rig to produce more appealing, camera-ready animations.

Finally, we only evaluate recordings of actors wearing a helmet-mounted camera. This setup ensures that the actor's head does not rotate relative to the camera. Avoiding head rotations simplifies the problem of style transfer and expression matching due to fewer degrees of freedom in the recordings. Future work could explore tracking head rotations so that actors will not need to wear a helmet-mounted camera.

Acknowledgements

We would like to thank Ian Spriggs for creating the facial blendshapes used in our work and Atri Dave for the animation used as training data. We further thank Sean Patrick Sherwin for helping with data capture, as well as creating the comparison animations in ARKit. Finally, we would like to thank all of the subjects who volunteered to be captured throughout the course of this project.

References

[BHB*11] BEELER T., HAHN F., BRADLEY D., BICKEL B., BEARDSLEY P., GOTSMAN C., SUMNER R. W., GROSS M.: High-quality passive facial performance capture using anchor frames. In *ACM SIGGRAPH 2011 papers*. 2011, pp. 1–10. 2, 11

[BRP*18] BOOTH J., ROUSSOS A., PONNIAH A., DUNAWAY D., ZAFEIRIOU S.: Large scale 3d morphable models. *Int. J. Comput. Vision 126*, 2–4 (Apr. 2018), 233–254. URL: https://doi.org/10.1007/s11263-017-1009-7, doi:10.1007/s11263-017-1009-7. 2

[BV99] BLANZ V., VETTER T.: A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 187–194. URL: https://doi.org/10.1145/311535.311556, doi:10.1145/311535.311556. 2

[BWP13] BOUAZIZ S., WANG Y., PAULY M.: Online modeling for realtime facial animation. *ACM Trans. Graph. 32*, 4 (July 2013), 40:1–40:10. URL: http://doi.acm.org/10.1145/2461912.2461976, doi:10.1145/2461912.2461976. 2

[CBGB20] CHANDRAN P., BRADLEY D., GROSS M., BEELER T.: Attention-driven cropping for very high resolution facial landmark detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 5861–5870. 5

[CCK*17] CHOI Y., CHOI M., KIM M., HA J., KIM S., CHOO J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *CoRR abs/1711.09020* (2017). URL: http://arxiv.org/abs/1711.09020, arXiv:1711.09020. 3

[CLY18] CAO K., LIAO J., YUAN L.: Carigans: Unpaired photo-to-caricature translation, 2018. 3

[CUYH20] CHOI Y., UH Y., YOO J., HA J.-W.: Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 3

[CWZ*14] CAO C., WENG Y., ZHOU S., TONG Y., ZHOU K.: Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 413–425. URL: https://doi.org/10.1109/TVCG.2013.249, doi:10.1109/TVCG.2013.249. 2

[DBB*18] DINEV D., BEELER T., BRADLEY D., BÄCHER M., XU H., KAVAN L.: User-guided lip correction for facial performance capture. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 93–101. 11

[DHT*00] DEBEVEC P., HAWKINS T., TCHOU C., DUIKER H.-P., SAROKIN W., SAGAR M.: Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 145–156. URL: https://doi.org/10.1145/344779.344855, doi:10.1145/344779.344855. 2

[FKA*18] FENG Z.-H., KITTLER J., AWAIS M., HUBER P., WU X.-J.: Wing loss for robust facial landmark localisation with convolutional neural networks. doi:10.1109/CVPR.2018.00238. 8

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2414–2423. 3

[HMB*18] HENDLER D., MOSER L., BATTULWAR R., CORRAL D., CRAMER P., MILLER R., CLOUDSDALE R., ROBLE D.: Avengers: Capturing thanos's complex face. In *ACM SIGGRAPH 2018 Talks* (New York, NY, USA, 2018), SIGGRAPH '18, Association for Computing Machinery. URL: https://doi.org/10.1145/3214745.3214766, doi:10.1145/3214745.3214766. 2

[IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A.: Image-to-image translation with conditional adversarial networks. pp. 5967–5976. doi:10.1109/CVPR.2017.632. 3

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision* (2016). 3

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014). 7

[KGT*18] KIM H., GARRIDO P., TEWARI A., XU W., THIES J., NIESSNER M., PÉREZ P., RICHARDT C., ZOLLHÖFER M., THEOBALT C.: Deep video portraits. *ACM Trans. Graph. 37*, 4 (July 2018). URL: https://doi.org/10.1145/3197517.3201283, doi:10.1145/3197517.3201283. 2

[KLA19] KARRAS T., LAINE S., AILA T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 3

[LD20] LI S., DENG W.: Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing* (2020), 1–1. doi:10.1109/TAFFC.2020.2981446. 3

[LKA*17] LAINE S., KARRAS T., AILA T., HERVA A., SAITO S., YU R., LI H., LEHTINEN J.: Production-level facial performance capture using deep convolutional neural networks. SCA '17, Association for Computing Machinery. URL: https://doi.org/10.1145/3099564.3099581, doi:10.1145/3099564.3099581. 2

[LSSS18] LOMBARDI S., SARAGIH J., SIMON T., SHEIKH Y.: Deep appearance models for face rendering. *ACM Trans. Graph. 37*, 4 (July 2018). URL: https://doi.org/10.1145/3197517.3201401, doi:10.1145/3197517.3201401. 2

[LWP10] LI H., WEISE T., PAULY M.: Example-based facial rigging. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2010) 29*, 3 (July 2010). 2

[LYYB13] LI H., YU J., YE Y., BREGLER C.: Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph. 32*, 4 (July 2013), 42:1–42:10. URL: http://doi.acm.org/10.1145/2461912.2462019, doi:10.1145/2461912.2462019. 2

[MCW*21] MOSER L., CHIEN C., WILLIAMS M., SERRA J., HENDLER D., ROBLE D.: Semi-supervised video-driven facial animation transfer for production. *ACM Transactions on Graphics (TOG) 40*, 6 (2021), 1–18. 3

[MD19] MA L., DENG Z.: Real-time hierarchical facial performance capture. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2019), pp. 1–10. 2

[MHR17] MOSER L., HENDLER D., ROBLE D.: Masquerade: Fine-scale details for head-mounted camera motion capture data. In *ACM SIGGRAPH 2017 Talks* (New York, NY, USA, 2017), SIGGRAPH '17, Association for Computing Machinery. URL: https://doi.org/10.1145/3084363.3085086, doi:10.1145/3084363.3085086. 2

[SAT*16] SAGONAS C., ANTONAKOS E., TZIMIROPOULOS G., ZAFEIRIOU S., PANTIC M.: 300 faces in-the-wild challenge: database and results. *Image and Vision Computing 47* (01 2016). doi:10.1016/j.imavis.2016.01.002. 5

[SLT*19] SIAROHIN A., LATHUILIÈRE S., TULYAKOV S., RICCI E., SEBE N.: First order motion model for image animation. In *Conference on Neural Information Processing Systems (NeurIPS)* (December 2019). 2, 3, 4

[SWW*20] SCHWARTZ G., WEI S.-E., WANG T.-L., LOMBARDI S., SIMON T., SARAGIH J., SHEIKH Y.: The eyes have it: An integrated eye and face model for photorealistic facial animation. *ACM Trans. Graph. 39*, 4 (July 2020). URL: https://doi.org/10.1145/3386569.3392493, doi:10.1145/3386569.3392493. 2

[THMM17] TRAN A., HASSNER T., MASI I., MEDIONI G.: Regressing robust and discriminative 3d morphable models with a very deep neural network. pp. 1493–1502. doi:10.1109/CVPR.2017.163. 2

[TYL17] TRAN L., YIN X., LIU X.: Disentangled representation learning gan for pose-invariant face recognition. In *In Proceeding of IEEE Computer Vision and Pattern Recognition* (Honolulu, HI, July 2017). 3

[TZK*17] TEWARI A., ZOLLHOFER M., KIM H., GARRIDO P., BERNARD F., PEREZ P., THEOBALT C.: Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. pp. 1274–1283. doi:10.1109/ICCVW.2017.153. 2

[ULVL16] ULYANOV D., LEBEDEV V., VEDALDI A., LEMPITSKY V.: Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (2016), ICML'16, JMLR.org, pp. 1349–1357. 3

[UVL16] ULYANOV D., VEDALDI A., LEMPITSKY V. S.: Instance normalization: The missing ingredient for fast stylization. *CoRR abs/1607.08022* (2016). URL: http://arxiv.org/abs/1607.08022, arXiv:1607.08022. 3

[VA19] VEMULAPALLI R., AGARWALA A.: A compact embedding for facial expression similarity. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 5676–5685. 3

[WBLP11] WEISE T., BOUAZIZ S., LI H., PAULY M.: Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 4 (July 2011), 77:1–77:10. URL: http://doi.acm.org/10.1145/2010324.1964972, doi:10.1145/2010324.1964972. 2

[WKZ18] WILES O., KOEPKE A. S., ZISSERMAN A.: X2face: A network for controlling face generation using images, audio, and pose codes. In *Proceedings of the European Conference on Computer Vision (ECCV)* (September 2018). 3

[WSS*19] WEI S.-E., SARAGIH J., SIMON T., HARLEY A. W., LOMBARDI S., PERDOCH M., HYPES A., WANG D., BADINO H., SHEIKH Y.: Vr facial animation via multiview image translation. *ACM Trans. Graph. 38*, 4 (July 2019). URL: https://doi.org/10.1145/3306346.3323030, doi:10.1145/3306346.3323030. 2

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017). 3

[ZSBL19] ZAKHAROV E., SHYSHEYA A., BURKOV E., LEMPITSKY V.: Few-shot adversarial learning of realistic neural talking head models. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 9458–9467. 3

[ZTG*18] ZOLLHÖFER M., THIES J., GARRIDO P., BRADLEY D., BEELER T., PÉREZ P., STAMMINGER M., NIESSNER M., THEOBALT C.: State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications. *Computer Graphics Forum* (2018). doi:10.1111/cgf.13382. 2