Fully Automatic Blendshape Generation for Stylized Characters

Jingying Wang[*]
Shanghai Jiao Tong University

Yilin Qiu[*]
Shanghai Jiao Tong University

Keyu Chen
University of Science and Technology of China

Yu Ding
Virtual Human Group, Netease Fuxi AI Lab

Ye Pan[†]
Shanghai Jiao Tong University

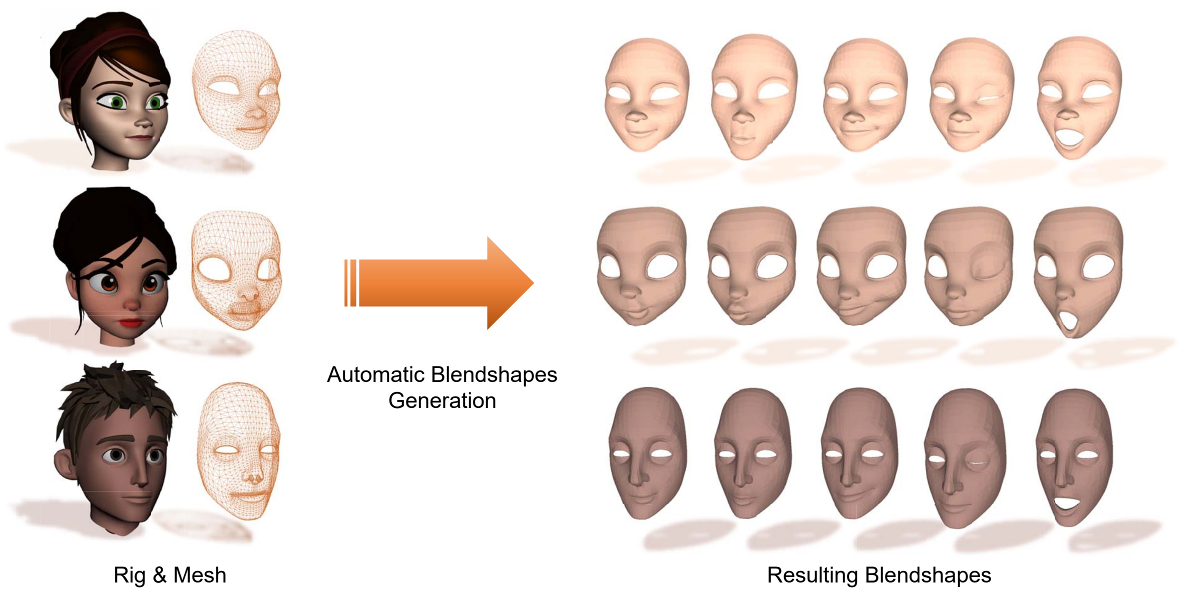


Figure 1: Automatically generated facial blendshapes for Stylized characters with different topologies.

Abstract

Avatars are one of the most important elements in virtual environments. Real-time facial retargeting technology is of vital importance in AR/VR interactions, the filmmaking, and the entertainment industry, and blendshapes for avatars are one of its important materials. Previous works either focused on the characters with the same topology, which cannot be generalized to universal avatars, or used optimization methods that have high demand on the dataset. In this paper, we adopt the essence of deep learning and feature transfer to realize deformation transfer, thereby generating blendshapes for target avatars based on the given sources. We proposed a Variational Autoencoder (VAE) to extract the latent space of the avatars and then use a Multilayer Perceptron (MLP) model to realize the translation between the latent spaces of the source avatar and target avatars. By decoding the latent code of different blendshapes, we can obtain the blendshapes for the target avatars with the same semantics as that of the source. We qualitatively and quantitatively compared our method with both classical and learning-based methods. The results revealed that the blendshapes generated by our method achieves higher similarity to the groundtruth blendshapes than the state-of-art methods. We also demonstrated that our method can be applied to expression transfer for stylized characters with different topologies.

[*]These authors contributed equally to this work.
[†]Corresponding author. The work is supervised by Dr. Ye Pan. E-mail: whitneypanye@sjtu.edu.cn.

Keywords: Avatars, Blendshapes, facial animation, stylized characters

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Real-time facial animation and retargeting is an essential topic in virtual environments and 3D modeling, having broad application in AR/VR interactions, social privacy, the filmmaking industry, and entertainment [8,21,27,28]. With the increasing population flooding in the Metaverse, this field is now moving towards increased flexibility to enable users to customize their digital avatars with the aim of increasing the accessibility of the technology to any user [13,23]. In the industry, a facial expression is often treated as a superposition of the movement of different muscles. Therefore, an efficient and reasonable way of driving avatars' faces using representatives called blendshapes has been invented. Each blendshape represents a modeled single action unit of the face with respect to the neutral face. The commercially available facial retargeting product of ARKit [2] and Faceware [9] calculate the weights of blendshapes, making it possible to recover the facial expression on an avatar by linearly combining the blendshapes [15].

Currently, the facial rigging and the generation of blendshapes for avatars strongly depend on expertise and human labor. Although some existing works have already explored how to automate this process, the user is still required to either install professional software such as MAYA and Blender or mark keypoints to facilitate the deformation transformation on computers, which still requires users' extra professional training and high quality of keypoint marking [7,17,22].

An automatic landmark detection can achieve high-quality keypoint marking for human faces but not for stylized characters. Some other works focused on the automatic rigging of human-like avatars that share the same model template [27], although this approach prevented users from employing other stylized characters. In this paper, we aim to propose a system that automatically generates a complete set of blendshapes for any given model without extra professions, including specialist software or expertise. With our system, users can perform real-time facial retargeting on their customized avatars using their smart phones.

We present a deep-learning-based method to morph the target avatar according to the deformation of the source avatar that represents the standard set of the blendshapes. The system consists of two variational autoencoders (VAEs) and a domain transfer network. Each VAE takes the deformed models as both the input and output. The output of the middle layer of the VAE is extracted as the latent space, and the domain transfer network maps the latent space of the source avatar to that of the target avatar to realize the deformation transformation. By taking advantage of the property of VAEs that the input and output are identical, the dataset required for online training only consists of a series of animations made by random sampling on the avatar's facial controllers. This feature is a further step in the overall automation and generalization of our system.

We tested our system with five stylized avatars: Mery (©meryproject), Ray (©cgtarian), Qing (©zhuiguang), Rose (©callesantiago), and Malcolm (©animschool), and based on the results for these characters, we assessed the our system both quantitatively and qualitatively. We sculpted blendshapes using an in-house artist as the groundtruth. In a comparison with a state-of-art methods [11, 20, 29], we found that the blendshapes generated by our method were much closer to the groundtruth. Our method outperformed [11, 20, 29] for blendshapes including "eyeBlink" and "jawOpen", "mouthRight", "mouthSmileLeft", and "mouthRollUpper", and achieved similar performance for other blendshapes such as "eyeSquintRight" and "mouthPucker". Moreover, the blendshapes generated by our algorithm were free of issues, such as, deformation arises in irrelevant areas, including the mouth, jaw, and left eyelid, asymmetric deformations, unexpected deformation, and wrong semantics. We also conducted a user study in which participants assigned similarity scores for the blendshapes generated by our method and [11,20,29] to the input source blendshapes. Results revealed the blendshapes obtained with our method received highest similarity score. Overall, our algorithm is able to deliver both accurate and high-quality blendshapes.

In summary, this study has the three main contributions:

1. To the best of our knowledge, we presented the first fully automatic blendshapes generation method for stylized characters with different topologies. We devised a pipeline for generating reliable datasets of pairwise facial rigs with the same semantics.

2. We demonstrated that the blendshapes generated by our algorithm are accurate and of high quality, and its application to sources having different styles, such as different expressions.

3. We released a set of 52 standard blendshapes for eight characters sculpted our in-house artist. This dataset motivates the further study in the domains of 3D character animation.

2 Related Work

The blendshapes of a face model are a group of morphed versions of the face model [14]. The blendshapes share the same topology as the face model. They consist of some base expressions of the face, and a comprehensive expression is generated by linearly combining them. Mathematically, the neutral face model is denoted as b_0, and the the set of blendshapes is $\{b_1, b_2, ..., b_N\}$. The comprehensive facial expression of the model can be written as

$$f = b_0 + \sum_{i=1}^{N} w_i (b_i - b_0), \tag{1}$$

where w_i are the weights of the blendshapes. By calculating the values of the blendshape weights, different expressions can be obtained with little computation [14].

Each facial retargeting system has its standard definition of blendshapes; thus, the formulation of blendshapes often uses a group of source blendshapes as a reference. Some methods directly transfer the deformation of each face on the source mesh to the target mesh, some methods extract the blendshapes from compound expressions by optimization given blendshape weights, and some methods resolve blendshapes when the source and target share the same topology.

2.1 Deformation Transfer Methods

Deformation transfer can be achieved through two main ways: classical methods and learning-based methods. Classical methods are normally lightweight, but require manual labelling jobs [20]. Learning-based methods supports automatic deformation with trained models, but a large dataset is required [11, 29]. Both approaches will be introduced and discussed below.

2.1.1 Classical Methods

Noh and Neumann [19], and Sumner et al. [25] both first used labeled correspondence points of the source and target models to find the face-to-face correspondence between their neutral models, and then they transferred the motions of the vertices and surface according to the found correspondence. [19] employed Radial Basis Functions (RBF) and cylindrical projection to morph the source model to the target model and find the dense surface correspondences, whereas [25] calibrated the correspondence points to iteratively optimize the source to the target. [19] transferred the motion of the vertices, whereas [25] transferred that of the faces. Onizuka et al. [20] further extended [25] by adding the loss of the difference of movement of selected landmarks between the source and target to improve the performance.

2.1.2 Learning-based Methods

Some recent studies have shown that deep learning methods such as VAEs can transfer the deformation of one mesh to another mesh effectively and semantically [26] [11] [24] [29]. This means that deformation can be transferred between meshes with different topologies [11]. The VAE takes a series of morphed versions of 3D mesh as a training set. One VAE is trained for the source avatar, and the other is trained for the target avatar. The latent space S^k_{human} for the expression k is obtained by inputting the source avatar with this expression to the VAE. After mapping latent space S^k_{human} to S^k_{avatar}, the D_{avatar} is able to generate a morphed target model with the same semantic meaning as the source.

Our work was inspired by [11, 29], but it has differences in terms of task definition and takes one more step in automatic data construction and model design. The method mentioned in [11] was not developed for the semantic transfer of facial expression between avatars, particularly not for the generation of blendshapes containing only subtle deformation. Additionally, in the former study of [29], considerable effort was devoted to manually constructing hierarchical human-avatar expression correspondences and to training the human-to-avatar expression retargeting network based on those data.

2.2 Optimization Methods

Li et al. [15] proposed an alternative iterative optimization method. In Step A, blendshape weights were fixed to minimize the loss of the reconstructed expression in terms of blendshapes. In Step B, the blendshapes were fixed and optimized in terms of weights. The two steps were alternatively employed to achieve convergence. Neumann et al. [18] attempted to decompose compound expressions with sparser representations, which were blendshapes. This method is similar to that in [15], which alternatively minimized the loss function in terms of blendshapes and weights. Also inspired by [15], Li et al. [16] generated a rigged face model for a person from a single scan. They estimated the blendshapes for a personalized model from the source by minimizing the loss of reconstructed expressions given a set of captured expressions of the target and the corresponding blendshape weights.

2.3 Same Topology Problem

Bouaziz et al. [3] generated a personalized target model by performing training on a large set of facial meshes with the same topology to study its identity PCA model, thus identifying the deformation fields of template blendshapes. The blendshapes of target avatars were modeled as the sum of the product of transfer operator and neutral face of the target, and the deformation fields of the specific expression. Carrigan et al. [5] performed training on a set of expressions of the target model with given blendshape weights. Since the target and the source share the same topology, the expression training set of the target was generated from that of the source.

3 Methodology

This section describes the overall structure of our blendshape generation system. As shown in Fig.2, the structure includes two components: (1) the latent space extractor and (2) the latent space converter. The core idea of our method is to use the VAE to extract the latent space from both source model meshes and target model meshes, which are denoted as L_{source_θ} and L_{target_θ} respectively, and then use the latent space transfer network to convert between the two latent spaces. These processes are expressed by the following equations:

$$L_{source_\theta} = \mathscr{E}_{\text{source}}(\text{source mesh}), \tag{2}$$

$$L_{target_\theta} = \mathscr{F}(L_{source_\theta}), \tag{3}$$

$$\text{target mesh} = \mathscr{D}_{\text{target}}(L_{target_\theta}), \tag{4}$$

where $\mathscr{E}_{\text{source}}$ represents the encoder of the VAE for the source avatar, and $\mathscr{D}_{\text{target}}$ represents the decoder of the VAE for the target avatar. This method is based on the fact that the variational autoencoder can construct a linearly continuous latent space.

3.1 Latent Space Extractor

The robustness of the VAE in deformation transfer has been validated in multiple works [26] [11] [24] [29]. The structure of the VAE is depicted in Fig.2. The input and output of the VAE network are the same, which reduces the workload in labeling the dataset. VAE is trained with discrete samples of meshes, but it can construct a continuous and linear latent space. Even though we do not use the latent space coded from the training samples, we can still obtain a plausible output on the decoder side due to the characteristics of the VAE that the input is encoded as a distribution over the latent space. The input and output mesh features for the VAE are denoted as M and M', Z represents the learned latent space, and $\mathscr{E}$ and $\mathscr{D}$ are the encoder and decoder of the VAE respectively. Then the loss function can be defined by two components:

$$\mathscr{L}_{\text{rec}} = ||M - \mathscr{D}(\mathscr{E}(M))||_1 = ||M - \mathscr{D}(Z)||_1 = ||M - M'||_1, \tag{5}$$

$$\mathscr{L}_{\text{KL-Divergence}} = KL(p(Z)||q(Z|M)), \tag{6}$$

where $\mathscr{L}_{\text{rec}}$ is the direct loss between the input mesh and the reconstructed mesh. $\mathscr{L}_{\text{KL-Divergence}}$ is the Kullback-Leibler (KL)

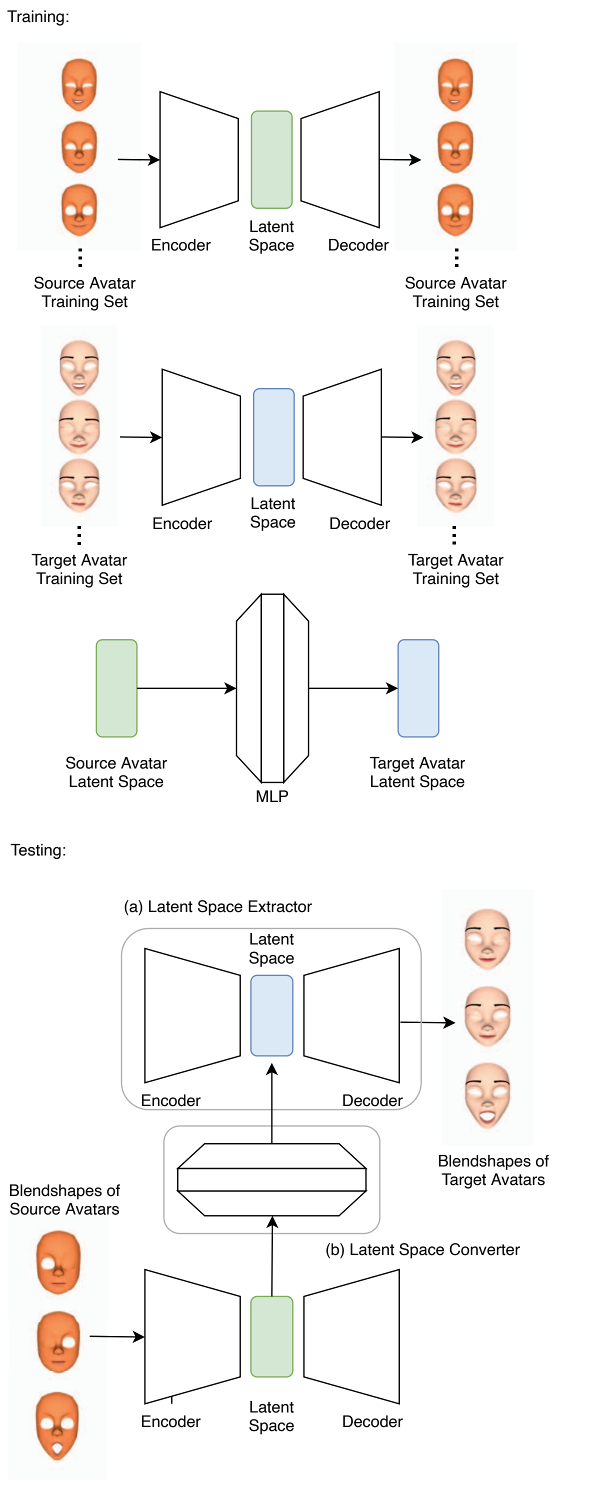

Training:
Encoder
Latent Space
Decoder
Source Avatar Training Set
Source Avatar Training Set
Encoder
Latent Space
Decoder
Target Avatar Training Set
Target Avatar Training Set
Source Avatar Latent Space
MLP
Target Avatar Latent Space
Testing:
(a) Latent Space Extractor
Latent Space
Encoder
Decoder
Blendshapes of Target Avatars
Blendshapes of Source Avatars
(b) Latent Space Converter
Encoder
Latent Space
Decoder

Figure 2: The overall workflow of proposed method. The training procedure of our system consists of two variational autoencoders and one multilayer perceptron. The testing pipeline starts from using the encoder of VAE to retrieve the latent code for blendshapes of source characters, and then uses MLP model to map it to the latent space of the target avatar, thus decoded to the target blendshapes.

divergence loss with posterior probability $q(Z|M)$ and a Gaussian distributed prior on the latent space Z, where

$$Z \sim \mathcal{N}(0,1). \tag{7}$$

The training process which is illustrated in Fig.2 shows that the training procedure of the two VAEs are independent of each other, and it is not even essential to train them on identically sized datasets to achieve any synchronization.

3.2 Latent Space Converter

To transfer the latent space coded from the blendshapes of the source avatars to replace that of the target avatars, a transfer function between the two latent spaces is constructed since the latent spaces of the two avatars do not necessarily have the same coordinates. The transfer function should output the latent code for the target avatar with the same geometric and semantic features as the source avatar. The supervised learning method is intuitive and effective for such a vector-to-vector conversion problem because it constructs a latent-code-to-latent-code dataset for both avatars. The dataset requires the same semantics for the source avatar model represented by the input latent code and the target avatar model represented by the output latent code to achieve synchronization. Therefore the multi-layer perceptron MLP model is suitable and effective for the translation. The loss for the latent space converter is constructed as,

$$\mathcal{L} = ||Z_{\text{target}} - \mathcal{F}(Z_{\text{source}})||_2, \tag{8}$$

where Z denotes the input latent code, and $\mathcal{F}$ denotes the latent space converter. This loss is measured as the mean square error between the output of the network and the groundtruth latent code of the target avatar.

As depicted in Fig.2, after the finishing the training of the two Latent Space Extractors, the training of the Latent Space Converter can be started.

4 Implementation

This section describes some implementation details, including how we establish the training dataset, how we computed the deformation representation for different models with the same topology, the structures of the VAE and MLP networks, and training parameters.

4.1 Dataset Construction

In this work, we used facially rigged models of five characters, Mery, Ray, Qing, Rose, and Malcolm. On each character's face, there are controllers embedded in the components such as eyelids, nose, lips, and jaw to emulate muscle movements. The generation of the dataset is based on the dataset published by Aneja et al. [1] which contains 10,000 expressions each for Bonnie (©joshsobelrigs), Mery, Ray, and Malcolm, as well as the random sampling of facial controller parameters. Furthermore, we also took 10,000 expressions/frames from movie clips for Qing and Rose. We clustered the controller semantically according to its distribution and location to ensure the plausibility of the randomly generated facial animation. For example, we clustered the three controllers on the left eyelid of the character and synchronized them. Therefore, for the training set of each VAE, we selected 480 expressions of each avatar from the dataset, and then we generated 4,320 expressions randomly.

The size of the training set of MLP Latent space converter is 900. The preparation of this dataset is completely automatic, and can be performed within a few seconds, which is a practical benefits of our proposed approach. The animated avatars contain thousands of frames, which we can easily obtained from 2-3 minutes of movie clips. We designed an algorithm to select animation pairs from source and target avatars that have matched expressions and geometries. The main strategy is that given an animated source avatar model, we search in the dataset of the target avatar to determine

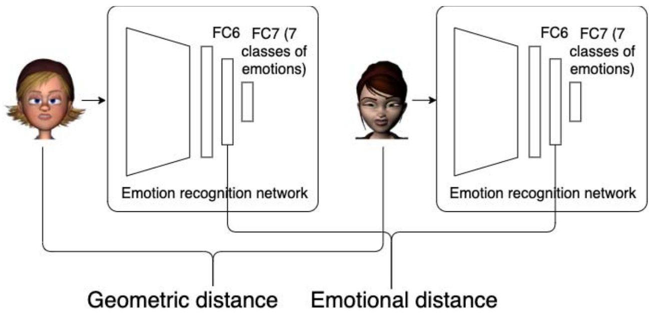

FC6
FC7 (7 classes of emotions)
Emotion recognition network
FC6
FC7 (7 classes of emotions)
Emotion recognition network
Geometric distance
Emotional distance

Figure 3: The strategy of forming facial rigs pairs.

whether there is an animation with sufficiently low "emotional distance" and "geometric distance" from the source as shown in Fig.3. If an animation of the target avatar that is emotionally and geometrically close to that of the source exists, then we include the pair with the lowest distance into the "paired dataset". To measure the emotional distance, we first rendered every frame of the animation in the dataset as an image. We then input the images to a trained neural network for emotion classification (into seven classes: angry, disgusted, fearful, happy, sad, neutral, and surprised). We retrieved the parameters of the last fully connected layer as the emotion vector of the model. The emotional distance between the two models is calculated as the Jensen-Shannon divergence of the two emotion vectors. The geometric distance is the sum of the distances between the 49 landmarks on the faces of the two avatars. We first searched for the 30 landmarks with the lowest emotional distance and found the ones with lowest geometric distance. A pair of facial rigs was established if the the two distances were jointly below a threshold. The emotional distance was first ranked for each animation of the source avatar to find the top 30 animations of the target that have the most similar emotion. We then ranked the geometric distance for the animation of the source avatar in the range of the 30 animation candidates of the target one, and select the one with the lowest geometric distance. If we cannot find an animation of the target avatar that has a geometric distance less than 4.9 after the normalization of the facial landmarks, we then abandon the construction of the pair for this source animation.

4.2 Feature extraction for Mesh

To input the meshes into the VAE model, it is necessary to fit them into a reasonable data structure. For the dataset of each character, the animations with different expressions are all morphed versions of its neutral phase; thus, the whole dataset shares the same topology. In this sense, we represent every expression of a character with respect to its neutral phase via the Deformation Representation (DR) feature [12] [10]. Each mesh in the dataset of a character is denoted as $M_k = (V_k, E)$ and the neutral phase is $M_0 = (V_0, E)$, where V_k represents the 3D positions of the vertices of the meshes and E represents the edges that connect the vertices.

The DR feature T_i^k for p_i^k, the i^{th} vertex of M_k is computed by minimizing the energy function,

$$E = \sum_{\text{neighbor vertex} j} c_{ij} ||\boldsymbol{e}_{ij}^k - T_k^i \boldsymbol{e}_{ij}^0||, \tag{9}$$

where $\boldsymbol{e}_{ij}^k = p_i^k - p_j^k$ is the vector edge of M_k and $\boldsymbol{e}_{ij}^0 = p_i^0 - p_j^0$ is the corresponding vector edge of M_0. Therefore, the DR feature T_i^k is the transformation matrix that represents the scaling and shearing of p_i^k's one ring neighbor. The computed representation is anine-dimensional vector at each vertex. All these vectors are then concatenated into a $|V| \times 9$ matrix.

4.3 Structure of VAE Latent Space Extractor

The structure of our proposed VAE is described in Fig.4 and Tab.1. The input l_0 and output l_7 of the network are both DRs of the 3D models with size $9 \times |V|$. The middle layers l_1, l_2, l_3, l_4, l_5 are all convolutional layers. l_6 is the reshape of l_5 from $1 \times |V|$ to $|V|$, which is followed by the fully connected layer F_1. The latent space is spanned by Z_2 with 25 dimensions.

4.4 Structure for MLP Latent Space Converter

The structure of our proposed MLP network is described in Fig.4 and Tab.2. The Latent Space Converter is a seven-layer MLP model. The input of the Latent Space Converter space is the latent code extracted from the blendshape models of the source avatar, which is a 25-dimensional vector. The output of the converter is that of the target avatar with the same format. The middle layers all have 100 dimensions.

4.5 Training Details

We have trained two deep learning models: the Latent Space Extractor and Latent Space Converter. We used Bonnie as the source avatar and five other different avatars, Mery, Ray, Qing, Rose, and Malcolm, as the target avatars. The training datasets of the two VAEs contain 4,800 expressions for the source avatar Bonnie and 4,800 expressions for each target avatar. All the VAE networks were trained for 80 epochs with a learning rate of 1e-4 and a decay of 0.5 every ten epochs. We set the batch size to 30 and the dimension of the latent space to 30. The training requires about two days.

The Latent Space Converter is the MLP model. The training dataset for this network consists of 900 pairs of latent codes for both the source and target avatars. All the MLP networks were trained for 50 epochs with a learning rate of 1e-4 and a decay of 0.6 every ten epochs. This model required about five hours to train.

The training frameworks were implemented in Keras 2.2.0 with Tensorflow 1.10.0 backend. All the implementations were run on a PC with CUDA 9.0. The training code and labeled dataset are available at `https://whitneypanye.github.io/`.

5 Evaluation

Our method was evaluated with three other competitive baselines, including geometric optimization and deep learning.

The first competitive baseline was the landmark-guided deformation transfer method (LDT) [20]. Following the implementation procedure of LDT, the 42 landmarks were manually labeled on both the source and target avatars in the neutral phase. Then the LDT was applied to generate the deformed blendshapes of target avatars with the source blendshapes.

Next, the automatic unpaired shape deformation transfer approach (Gao et al.) [11] was employed as the second baseline. Despite the original work being designed for general shape deformations, it was still comparable and applicable to the automatic blendshape generation problem. Specifically, the same VAE latent space extractor was used for our model as the baseline for automatic unpaired shape deformation transfer. But, the MLP latent space converter was replaced with a new one trained on the light field distance (LFD) metric [6].

The last baseline was the facial expression retargeting framework made by easy manual annotation (Zhang et al.) [29]. Similar to the automatic unpaired shape deformation transfer method, our model's VAE model was kept. As for the latent space converter, the triplet annotation pipeline for the facial expression retargeting framework was re-implemented and generated thousands of triple-wise data to re-train the MLP network.

For a fair comparison, the automatic unpaired shape deformation transfer method [11], the same network structure was kept for the facial expression retargeting framework [29], and our model.

5.1 Quantitative Analysis

The industrial standard for the facial blendshapes has been well defined by some commercially available products such as ARKit [2]. In this work, we created a set of blendshapes for each avatar by in-house artist as groundtruth, which are "eyeBlinkLeft", "eyeBlinkRight", "eyeSquintLeft", "eyeSquintRight", "jawLeft", "jawRight", "jawOpen", "mouthLeft", "mouthRight", "mouth smile left", "mouthSmileRight", "mouthDimpleLeft", "mouthDimpleRight", "mouthRollUpper", "browDownLeft", "browDownRight", "browOuterUpLeft", "browOuterUpRight", "mouthFunnel", and "mouthPucker". These blendshapes all follow the standard fashion defined by ARKit and have consistent semantics.

To quantitatively estimate the distance between our result and the groundtruth, and also that between the baseline result and the groundtruth, we calculated the mean square error between the results and the groundtruth. Mathematically, for each vertex r_i in the resulting blendshape, and vertex g_i in the groundtruth blendshape, the mean square error for the blendshape $b \in \{0, 1, 2, ..., 19\}$ was estimated as,

$$\mathrm{MSE}_b = \sum_{i=0}^{N} ||\boldsymbol{r_i^b}(x,y,z) - \boldsymbol{g_i^b}(x,y,z)||_2^2, \tag{10}$$

where $\boldsymbol{r_i^b}(x,y,z)$ and $\boldsymbol{g_i^b}(x,y,z)$ represent the 3D positions of the r_i^b and g_i^b respectively.

Figure 5 shows the mean square errors for four methods across 7 basic blendshapes. Our method achieved lowest error score ($Mean, M = .085$, $StandardError, SE = .01$), demonstrating the effectiveness of our algorithm. The mean error increased from the our algorithm to Zhang et al. ($M = .144, SE = .023$), to LDT ($M = .153, SE = .008$) to Gao et al. ($M = .181, SE = .129$) methods, in that order.

A one-way ANOVA was conducted to determine the effects of different methods on mean square error. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro-Wilk test ($p > .05$). There was homogeneity of variances, as assessed by Levene's test for equality of variances, $p = .147$. There was a significant main effect of different methods, $F(3,139) = 5.654, p = .001$. Tukey post hoc analysis revealed that the mean error for our method was significantly lower than the LDT ($p = .005$), the Zhang et al. ($p = .016$) and the Gao et al., ($p < .001$), and no other group differences were statistically significant.

5.2 User Study

The aim of the user study was to verify that the blendshapes generated by our algorithm are visually and semantically correct.

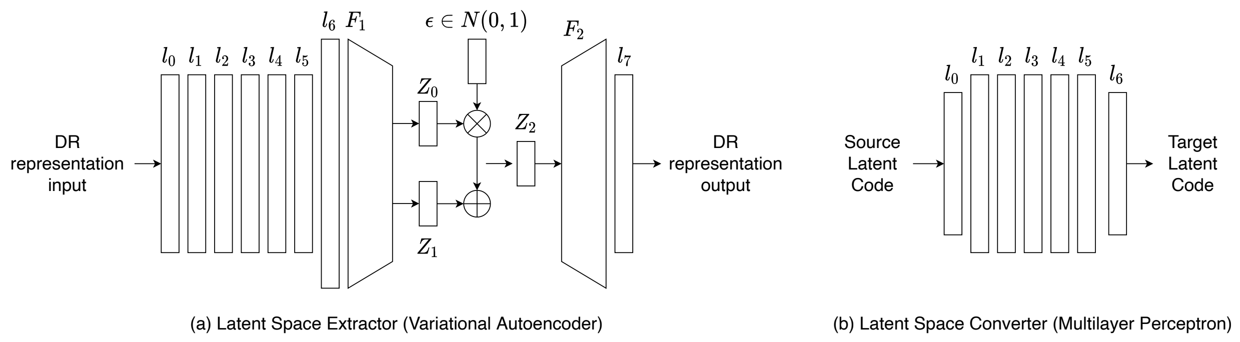
5.2.1 Participants & design

We recruited 30 participants (14 males and 16 females) to complete the survey. The participants were all college students aged 18 to 25 years old. They were naïve to the purposes of the experiment.

The experiment utilized 4 methods (Ours, LDT, Gao & Zhang) $\times$ characters 5 (Mery, Ray, Qing, Rose & Malcolm) $\times$ 7 blendshapes in a within-subject design. Thus, each participant took part in 140 trials in total. To avoid fatigue or carry-over effects, images were viewed by the participants in random order.

5.2.2 Task & Procedures

The participants all signed the consent form before engaging in the trial. The participants were given a pair of blendshapes, which is the source and target blendshapes, together with the blendshape name. They assessed the similarity of the paired blendshapes, and the semantic correspondence of the target blendshape to its name on



l_0 l_1 l_2 l_3 l_4 l_5 l_6 F_1
$\epsilon \in N(0,1)$
Z_0 Z_1 Z_2 F_2 l_7
DR representation input
DR representation output
Source Latent Code
Target Latent Code
(a) Latent Space Extractor (Variational Autoencoder)
(b) Latent Space Converter (Multilayer Perceptron)

Figure 4: The structure of the proposed networks.

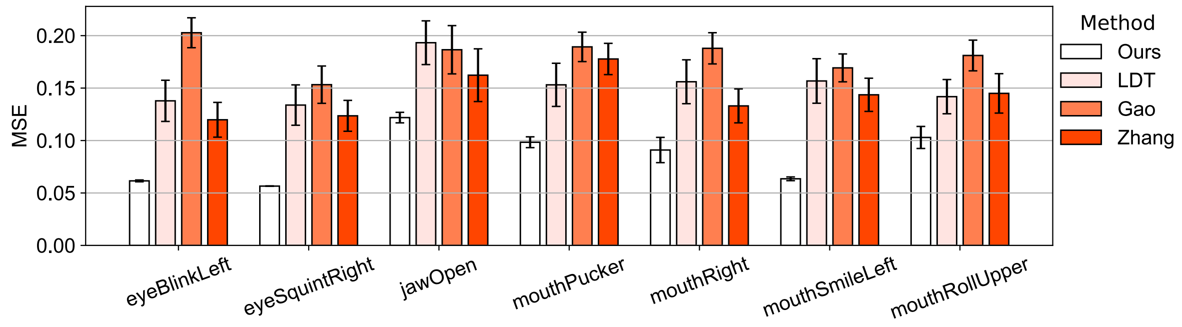


Figure 5: Statistics of quantitative study. Mean square error of results generated by [11, 20, 29] and our method respectively versus 7 basic blendshapes.

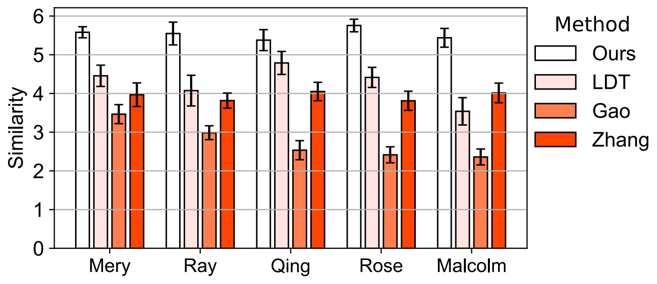


Figure 6: Statistics of the user study. The bars demonstrate the mean and standard deviation of the received score of how similar the results generated by [11, 20, 29] and our method for Mery, Ray, Qing, Rose, and Malcolm, respectively, are to the source blendshapes.

a seven-point likert scale. The content of the survey can be accessed at `https://forms.gle/K5pbKFkcLCEcLFe59`. The survey contained 142 questions. Three of them were demographic questions, 140 questions asked participants to evaluate the similarity of the source and the target generated by difference methods.

The survey required about 10 minutes to complete. The participants were paid 7 USD. The experiment was approved by Shanghai Jiao Tong University Research Ethics Committee.

5.2.3 Results

Figure 6 shows the average similarity scores obtained for four methods across five characters. Our method achieved highest similarity score $(Mean, M = 5.528,\ StandardError, SE = .125)$, demonstrating the effectiveness of our algorithm. The mean score decreased from the our algorithm to LDT $(M = 4.203, SE = .135)$, to Zhang et al. $(M = 3.895, SE = .121)$ to Gao et al. $(M = 2.713, SE = .111)$ methods, in that order.

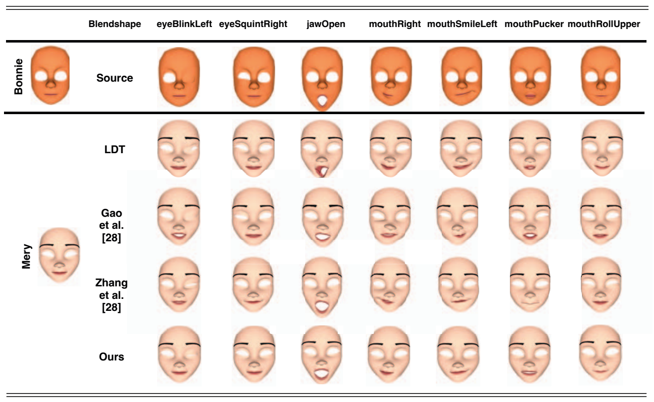
A three-way repeated measures ANOVA was conducted to determine the effects of different methods on similarity score. There were no outliers, as assessed by boxplot; data was normally distributed for each group, as assessed by Shapiro-Wilk test $(p > .05)$. The assumption of sphericity was violated, as assessed by Mauchly's test of sphericity, $\chi^2(5) = 29.777, p < .001$. Therefore, a Greenhouse-Geisser correction was applied. There was a significant main effect of different methods, $F(1.942, 56.316) = 169.552, p < .001$. Post hoc analysis with a Bonferroni adjustment revealed that the mean similarity score for our method was significantly higher than the LDT $(p < .001)$, the Zhang et al. $(p < .001)$ and the Gao et al., $(p < .001)$, respectively.

6 Discussion

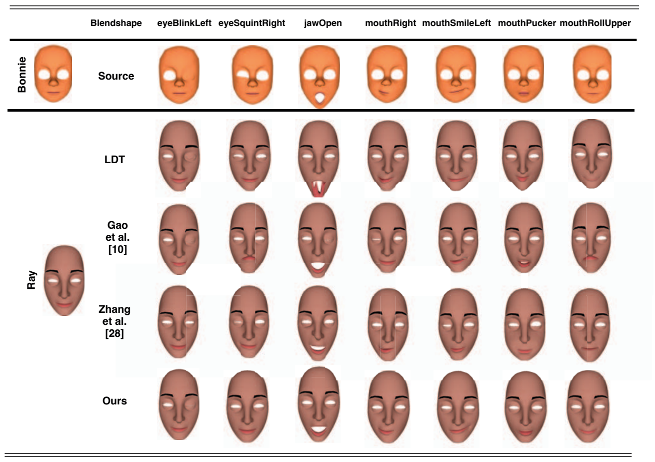
In this section, we discussed findings obtained from the experiment and its applications. First, we discussed results comparing with the baseline methods. Second, in our experiment, we employed multiple characters as source avatars; in addition to Bonnie, we showed human characters can also be a good source of the stylized target avatars. Third, our system was found to also to be robust in directly transferring emotional expressions. Lastly, by integrating our generated blendshapes of Mery and the ARKit Facial Retargeting function for calculating blendshape weights, end users can generate facial animations with their iPhone devices.

6.1 Discussion on results

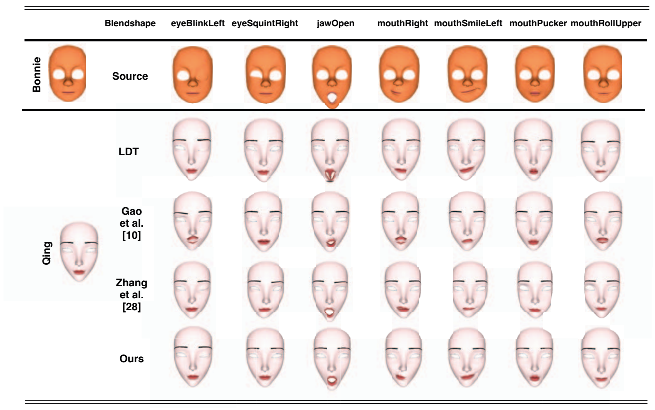
Figure 7 presents the different blendshapes generated by four methods for five characters. We observed that the LDT method might produce rough edges, while the automatic unpaired shape deformation transfer methods used by Gao et al. [11] and the facial expression



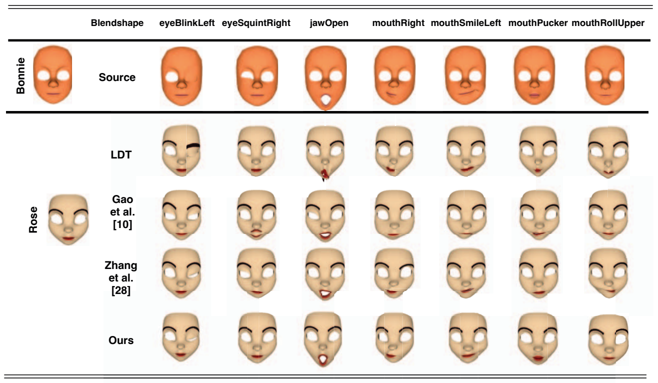
(a) 7 Blendshapes generated for Mery.



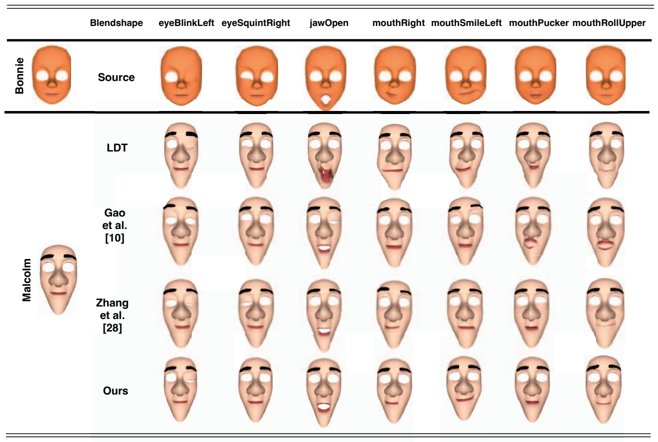
(b) 7 Blendshapes generated for Ray.



(c) 7 Blendshapes generated for Qing.



(d) 7 Blendshapes generated for Rose.



(e) 7 Blendshapes generated for Malcom.

Figure 7: Comparison of blendshape generation for five avatars: Mery, Ray, Qing, Rose, and Malcolm. The first row is the 7 basic source blendshapes, followed by results generated from [11, 20, 29] and our methods for Mery, Ray, Qing, Rose, and Malcolm respectively.

retargeting framework used by Zhang et al. [29], also produce an array of issues. Some of these issues include deformation arises in irrelevant areas, asymmetric deformations, corrupted deformation, and wrong semantics. As such, our method is robust in most cases and preserves symmetry and semantics well.

Obtaining some blendshapes have proven to be a difficult task for all the baseline methods, and each of these methods may have limitations. For example, The blendshape “eyeBlinkLeft” creates significant issues when the LTD method is applied in its creation. One of the issues is that the upper eyelid does not extend to the lower eyelid while the edges of the upper eyelid are not smooth. The blendshapes obtained by Gao et al. [11] and Zhang et al. [29] also have issues, including a deformed mouth instead of deformed eyelids. We note the upper eyelid generated by our method sometimes cannot quite reach the lower eyelid. Additionally, the application of the LTD method produces a high-quality blendshape “eyeSquintRight” with no issues present. However, when obtained through the Gao et al. [11] and the Zhang et al. [29], deformation arises in irrelevant areas, namely, the mouth. Our method strives to create blendshapes with minimal deformation of the lower eyelid. We note that the MSE for LDT method is still larger than our method, since MSE distance is a global metric, which might not reflect the accurate expression semantics, especially for minor facial deformations such

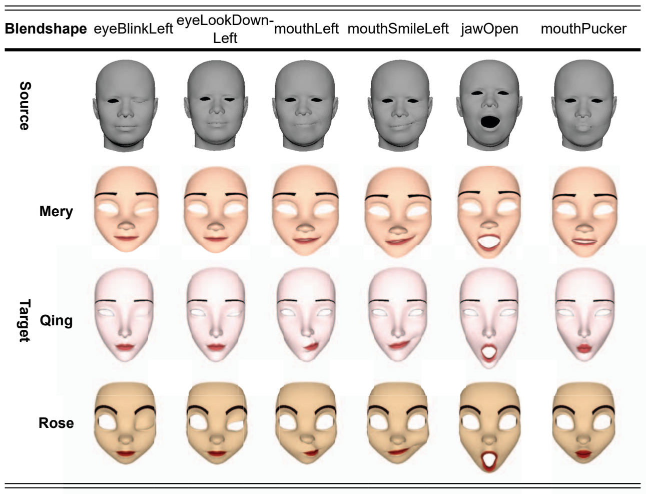


Figure 8: Examples for identity transfer results. The top row is source human face and its blendshapes, and the subsequent row is different character face and its automatically generated blendshapes.

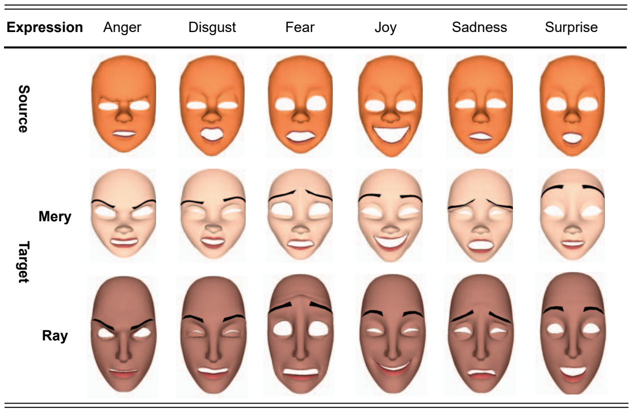


Figure 9: Examples for expression transfer results. The top row is source character face and its expressions, and subsequent row is target character faces and its automatically generated expressions.

as "eyeSquint".

From the qualitative and quantitative results demonstrated in Figure 5 to Figure 7, it can be observed that our method generated the most satisfactory target blendshapes regarding the expression semantics and facial deformations. In comparison, the LDT approach produced practical expressions in most cases but failed when upper and lower lip landmarks were mistakenly annotated. Thus, the blendshape generation quality of the LDT method was heavily dependent on the landmark annotation accuracy. Additionally, manually labeling landmarks for stylized characters requires a tremendous amount of time. There are OpenCV or other open-source frameworks can automatically extract landmarks for human face. However, these frameworks are difficult to apply to stylized characters with exaggerated and artistic expressions. On the other side, the deep learning-based methods, which include the automatic unpaired shape deformation transfer methods and the facial expression retargeting framework, were inspired by the data-driven idea. They required no proficient manual expertise, but these two baselines are not developed for transfer of blendshapes between avatars, which only contain subtle deformation. The automatic unpaired shape deformation transfer method had to measure the LFD [6] for optimizing the blendshape results. However, the LFD metric was too coarse to capture the fine details of facial deformations, such that the translated expression was generally missed in accuracy. Besides, the facial retargeting framework built upon easy annotations still needs at least hours from novice annotators. Despite its quite intensive labor requirements, the blendshape generation results of the facial expression retargeting framework may fail in specific cases like mouth puckering and smiling, which are less annotated in training data.

6.2 Blendshape Generation with Different Sources

In addition to using the stylized character Bonnie as the source avatar, it is also feasible that we use some other more realistic characters such as humans [4] as the source. First, Fig.8 illustrates the results of several key blendshapes, revealing that with a human character as the source, high-quality blendshapes were generated for Mery. However, using Bonnie as the input has the advantages of higher quality and more consistent styles. For the blendshape "eyeBlink", Bonnie-sourced blendshapes had more closed eyelids. For the blendshape "mouthSmile", Bonnie-sourced blendshapes had more exaggerated expressions. Second, Fig.8 also demonstrates our method can fully automatically generate a set of blendshapes for any arbitrary characters with different topologies.

6.3 Expression transfer

In addition to blendshape generation, our method is also effective in accurately transferring expressions with different emotions. We used Bonnie with six source expressions, anger, disgust, fear, joy, sadness, and surprise as the source model. The source model are all in different emotions. As we can infer from Fig.9 that the result of translation of expression is robust since both Mery and Ray demonstrate that the correct displacement of the components such as eyebrows, eyelids, nose, cheek, and lips; thus, the emotions are accurately conveyed.

6.4 Applications & future work

We created 52 blendshapes for the source character in the standard manner defined by ARKit. This resulted in a fully corresponding set of blendshapes between the source and target characters. Once the resulting blendshapes are embeded on an iPhone, the users will be able to enjoy a real-time facial retargeting service. We will also release the resulting blendshapes for 8 new characters, which will motivate the further study in the domains of character rigging and animation.

There are many potential avenues for next steps. To further improve the fidelity of our generated result, we could include several interesting methods (e.g. Laplacian [3] and Radial Basis Functions [19]). A potential route would be design these into the loss terms. We also hope to extend the evaluation to using more complex geometry with rich curvatures and details, e.g., small wrinkles. It would be interesting to validate if those details can be preserved after deformation. Lastly, we focus on facial gestures in this study. To support our method to be applicable to a generic scenario, we plan to include textures in future work.

7 Conclusion

To move one step further toward the automation of facial interactions in AR/VR, we proposed to use a deep learning method to generate blendshapes for avatars. By taking the advantage of the capability of the VAE of learning a smooth latent space, we constructed a latent space extractor as well as an MLP-model-based converter to bridge the latent space between the source and target avatars. By both quantitative and qualitative assessment, we can concluded that our method outperforms the traditional LDT transfer method in terms of quality and fidelity for some particular blendshapes. We will continue to refine the details of the blendshapes as well as to make the preparation and training process more accessible for public use.

Acknowledgments

This work was supported by Shanghai Sailing Program (No. 20YF1421200), National Natural Science Foundation of China (NSFC, NO. 62102255) and Hangzhou Key Science and Technology Innovation Program (No. 2022AIZD0054).

References

[1] D. Aneja, B. Chaudhuri, A. Colburn, G. Faigin, L. Shapiro, and B. Mones. Learning to generate 3d stylized character expressions from humans. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 160–169. IEEE, 2018.

[2] Apple. Tracking and visualizing faces, 2022.

[3] S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for realtime facial animation. *ACM Trans. Graph.*, 32(4), July 2013. doi: 10.1145/2461912.2461976

[4] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

[5] E. Carrigan, E. Zell, C. Guiard, and R. McDonnell. Expression packing: As-few-as-possible training expressions for blendshape transfer. *Computer Graphics Forum*, 39(2):219–233, 2020. doi: 10.1111/cgf.13925

[6] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, vol. 22, pp. 223–232. Wiley Online Library, 2003.

[7] F. Danieau, I. Gubins, N. Olivier, O. Dumas, B. Denis, T. Lopez, N. Mollet, B. Frager, and Q. Avril. Automatic generation and stylization of 3d facial rigs. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 784–792. IEEE, 2019.

[8] B. Egger, W. A. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39(5):1–38, 2020.

[9] Faceware. Intro to faceware studio, 2022.

[10] L. Gao, Y.-K. Lai, J. Yang, L.-X. Zhang, S. Xia, and L. Kobbelt. Sparse data driven mesh deformation. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2085–2100, 2021. doi: 10.1109/TVCG.2019.2941200

[11] L. Gao, J. Yang, Y.-L. Qiao, Y.-K. Lai, P. L. Rosin, W. Xu, and S. Xia. Automatic unpaired shape deformation transfer. *ACM Trans. Graph.*, 37(6), Dec. 2018. doi: 10.1145/3272127.3275028

[12] Z.-H. Jiang, Q. Wu, K. Chen, and J. Zhang. Disentangled representation learning for 3d face shape. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11949–11958, 2019. doi: 10.1109/CVPR.2019.01223

[13] J. Kim and K. Singh. Optimizing ui layouts for deformable face-rig manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021.

[14] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng. Practice and Theory of Blendshape Facial Models. In S. Lefebvre and M. Spagnuolo, eds., *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. doi: 10.2312/egst.20141042

[15] H. Li, T. Weise, and M. Pauly. Example-based facial rigging. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2010)*, 29(3), July 2010.

[16] J. Li, Z. Kuang, Y. Zhao, M. He, K. Bladin, and H. Li. Dynamic facial asset and rig generation from a single scan. *ACM Trans. Graph.*, 39(6), Nov. 2020. doi: 10.1145/3414685.3417817

[17] V. McGowen and J. Geigel. Automatic blend shape creation for facial motion capture. In *ACM SIGGRAPH 2016 Posters*, pp. 1–2. 2016.

[18] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. Sparse localized deformation components. *ACM Trans. Graph.*, 32(6), Nov. 2013. doi: 10.1145/2508363.2508417

[19] J.-y. Noh and U. Neumann. Expression cloning. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, p. 277–288. Association for Computing Machinery, New York, NY, USA, 2001. doi: 10.1145/383259.383290

[20] H. Onizuka, D. Thomas, H. Uchiyama, and R. Taniguchi. Landmark-guided deformation transfer of template facial expressions for automatic generation of avatar blendshapes. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2100–2108, 2019. doi: 10.1109/ICCVW.2019.00265

[21] J. Osipa. *Stop staring: facial modeling and animation done right*. John Wiley & Sons, 2010.

[22] C. Pawaskar, W.-C. Ma, K. Carnegie, J. P. Lewis, and T. Rhee. Expression transfer: A system to build 3d blend shapes for facial animation. In *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*, pp. 154–159. IEEE, 2013.

[23] Polywink. Automatic expressions blendshapes and facial rigs. 2021.

[24] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[25] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, Aug. 2004. doi: 10.1145/1015706.1015736

[26] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] M. Volonte, E. Ofek, K. Jakubzak, S. Bruner, and M. Gonzalez-Franco. Headbox: A facial blendshape animation toolkit for the microsoft rocketbox library. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 39–42. IEEE, 2022.

[28] Z. Wang, J. Ling, C. Feng, M. Lu, and F. Xu. Emotion-preserving blendshape update with real-time face tracking. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[29] J. Zhang, K. Chen, and J. Zheng. Facial expression retargeting from human to avatar made easy. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2020. doi: 10.1109/TVCG.2020.3013876

A Architecture of the Networks

layer	size
l_0 (input)	$9 \times \lvert V \rvert$
l_1 (Conv)	$128 \times \lvert V \rvert$
l_2 (Conv)	$64 \times \lvert V \rvert$
l_3 (Conv)	$64 \times \lvert V \rvert$
l_4 (Conv)	$64 \times \lvert V \rvert$
l_5 (Conv)	$1 \times \lvert V \rvert$
l_6 (Reshape)	$\lvert V \rvert$
F_1 (FC)	300
Z_0, Z_1, Z_2	25
F_2 (FC)	300
l_7 (output)	$9 \times \lvert V \rvert$

Table 1: Latent Space Extractor (Variational Autoencoder) architecture.

layer	size
l_0 (input)	25
l_1	100
l_2	100
l_3	100
l_4	100
l_5	100
l_6 (output)	25

Table 2: Latent Space Extractor (Multilayer Perceptron) architecture.