# [Supplementary] Controllable 3D Generative Adversarial Face Model via Disentangling Shape and Appearance

## Appendix 1

**Expression generator conditioned on identity:** A commonly acknowledged fact about human faces is each individual's ability to express themselves uniquely with differences in the fine details. The expression generator when trained with inputs as ($z_{noise}$, $z_{exp}$) can generate only a single style of expression for each identity. By using the $G_{exp}$ conditioned on $z_{id}$, our model can modify the style of expression associated with a shape's identity by changing only the $z_{id}$ code in $G_{exp}$, thereby increasing the overall diversity. As illustrated in Fig. 1(a), the same identity has the "smile" expression in various styles. In Fig. 1(b), the source identity has a "mouth right" expression, and the target has the same expression with a slight smile and deeper lip corner, which is transferred onto the source shape.

**Visualization of generated samples:** The embedding space of generated samples with the same level of expressions is futher explored. Specifically, we conduct t-SNE on 5000 samples generated by our model, including 250 different identities with 20 different expressions each. As shown in Fig. 2(a), the samples with the same identity are within the same cluster. Fig. 2(b), while, shows that samples with the same expression are within the same cluster.

We also explore the embedding space of generated samples with different levels of expressions. 3000 samples are generated including 15 different levels for all 20 expressions, and each level of a specific expression has 10 samples. The intensity of expression varies from 0.0 to 1.5 (higher values of number mean a higher level of intensity of expression). As shown in Fig. 3(a) by t-SNE, all the clusters can be roughly divided into two categories, i.e., ones with pure colors and others with mixed colors. Each color represents a specific expression. We randomly zoom in a cluster with pure color (region A) and the expression levels are primarily high values (above 0.5). We observed that samples close to the cluster's center have higher values (the most immediate samples have the highest values of 1.5). This is because samples with higher levels or intensities visually have more prominent expressions and they are more likely to belong to that corresponding expressions, and so does their embedding space as in Fig. 3(b). The samples with lower levels of expressions, on the other hand, are more like 'neutral' expression even though they belong to different expressions. As shown in Fig. 3(c), we randomly zoom in a cluster with mixed color (region B), and the expression levels are mostly low values (lower than 0.5). Besides, the more samples are close to the cluster's center, the lower values they have (the extreme case is that samples in the center have the lowest values, i.e., 0.0).

This set of experiments shows that our generative model can not only generate different levels of expressions visually but also make sense semantically in the corresponding embedding space. And our model indeed encodes identity and expression information in an implicitly meaningful way.

## Appendix 2

Here we present the results when the AUs are incorrect are shown in Fig. 4. We can observe that GANimation results hardly change since, with the wrong facial landmarks, the model regards the value of action units corresponding to the mouth region as nearly a constant from the source to the target. In this way, the images will not change no matter we perform interpolation or extrapolation.
In conclusion, our method has superior expression intensity control compared with GANimation.

**Architecture details:** In table 1 and table 2 we detail the architecture of our encoder and decoder. Here, $num\_verts$ denotes the number of a shape's vertices within a dataset and $num\_verts \times 3$ means the dimension of the flattened vector comprising all vertices' 3D coordinates $(x, y, z)$. Besides, $n_{id}$ and $n_{exp}$ are the number of identities and expressions within a dataset, respectively. In particular, $num\_verts = 26317$, $n_{id} = 847$ and $n_{exp} = 20$ for FaceScape dataset and $num\_verts = 22127$, $n_{id} = 267$ and $n_{exp} = 7$ for Comb dataset. In table 3 and table 4, we detail the architecture of our generator and discriminator. Identity code $z_{id}$ and noise code $z_{noise}$ are sampled from Gaussian distribution and expression code is a one-hot vector. We use LeakyReLU with a slope of 0.2 as our activation function.

## Appendix 3

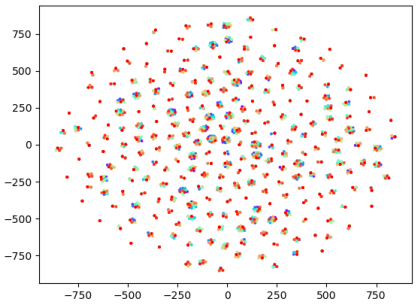
Fig. 5 illustrates extrapolation along the textures of randomly generated identities. By fixing the expression code, we can fix the expression and change the identity by varying the identity code and vice versa. Fig. 6 shows the rendered faces with the corresponding meshes for a given ID and different expressions.

We also trained our model on BP4D dataset - a highly diverse dataset. The dataset are labeled by AUs and tasks instead of holistic expression labels. However, for each of the basic facial expressions, we can obtain their prototypical AUs based on the facial action coding system (FACS) [2]. For example, the work [1] has done a study and provided a set of weights which indicates the percentage of the specific expression in which the AU are activated. Following the same model, we used the same weights to annotate the images with different facial expressions. For example, happiness is represented (see Table 1 in [1]) by $AU_{12}$, $AU_{25}$, $AU_6$, $AU_7$7, $AU_{10}$, with the corresponding weights of $0.82$, $0.7$, $0.57$, $0.83$, $0.63$, respectively. Here, for each video frame in the BP4D dataset, we calculate the level of each expression and pick the frame which gives us the maximum score for that expression.

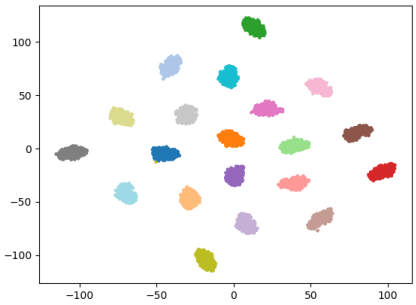
(a) Same identity having different styles of "smile" expression.


(b) Transferring expression style of target to source.

Figure 1. Style editing: changing only the $z_{id}$ code in $G_{exp}$ , the style of expression or fine details can be transferred while preserving the identity.




(a)



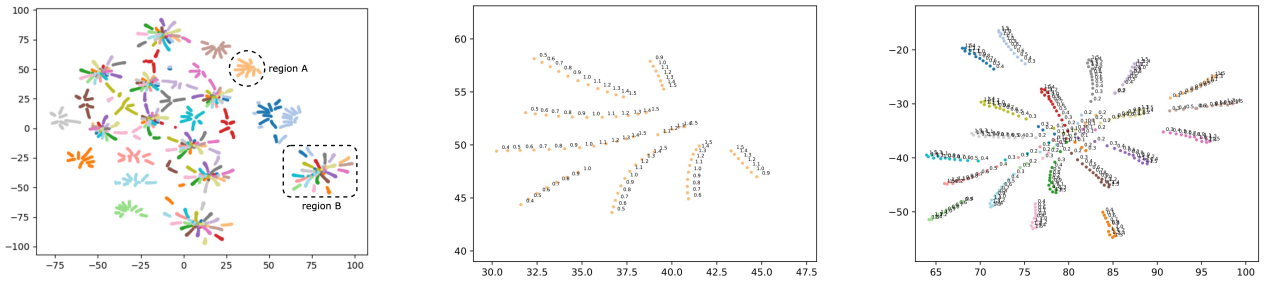

(b)

Figure 2. (a) and (b) are visualization results of identity and expression embedding respectively of generated samples with same level of expressions using t-SNE. There are 20 colors in each figure indicating the 20 expressions.

## References

[1] Dimitrios Kollias, Viktoriia Sharmanska, and Stefanos Zafeiriou. Distribution matching for heterogeneous multitask learning: a large-scale face study. *arXiv preprint arXiv:2105.03790*, 2021.

[2] Erika L Rosenberg and Paul Ekman. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 2020.




(a) (b) (c)

Figure 3. (a) visualization of expression embedding of generated samples with different levels of expressions using t-SNE. (b) zoom of region A of (a) and (c) zoom of region B of (a). The numbers in (b) and (c) denote the different levels or intensities of expressions.

Figure 4. The same setting as the last Figure. Mouth stretch on sample 8.

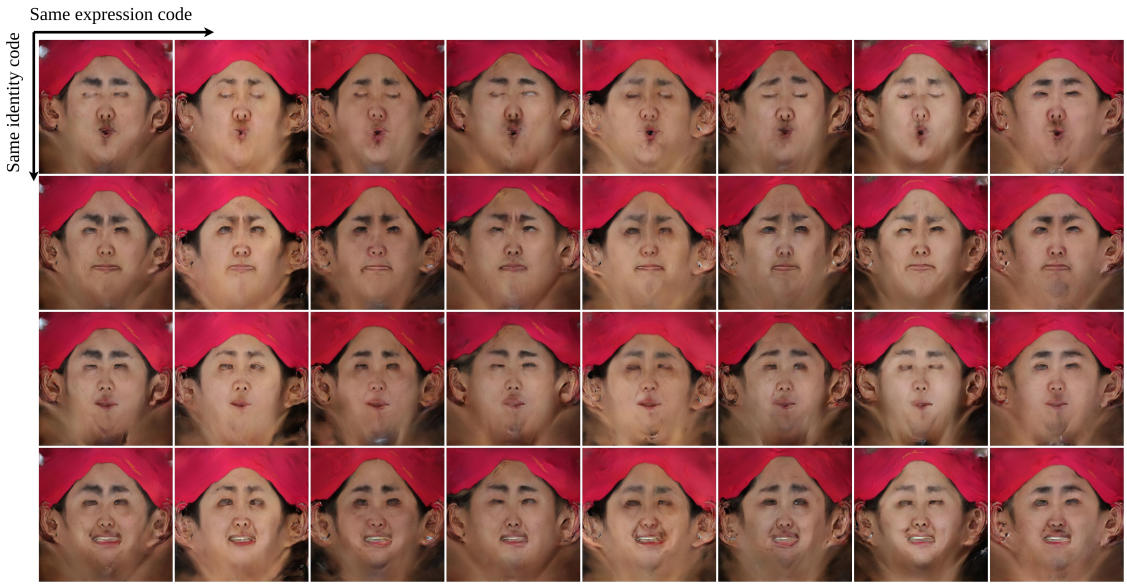



Figure 5. Generated Textures.

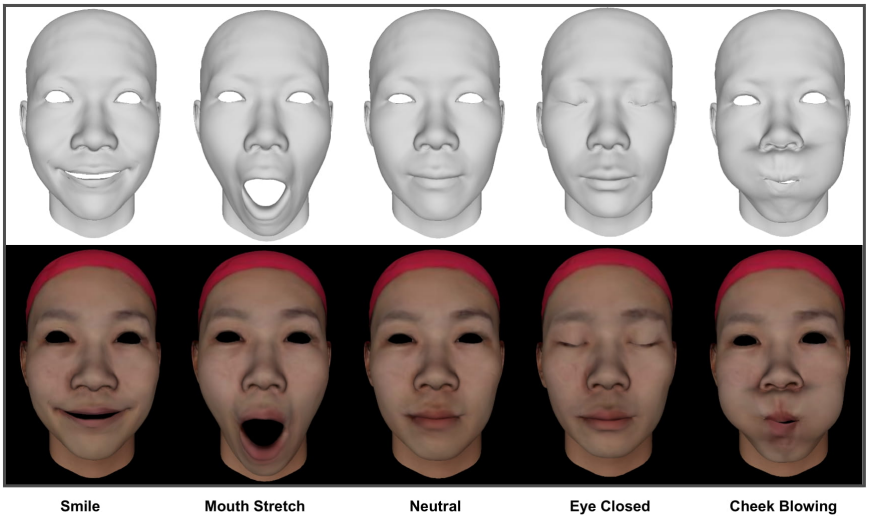



Figure 6. Meshes and rendered images of same ID, different expressions. The first row shows meshes generated using our shape generator and the second row shows the corresponding rendered images using the textures from the texture generator.

Table 1. The architecture of encoder

| | ID encoder | Exp encoder |
|---|---|---|
| Input | $num\_verts \times 3$ | $num\_verts \times 3$ |
| Linear | $num\_verts \times 3 \rightarrow 1024$ | $num\_verts \times 3 \rightarrow 1024$ |
| Activation | LeakyReLU(0.2) | |
| Linear | 1024→512 | 1024→512 |
| Activation | LeakyReLU(0.2) | |
| Linear | 512→100 | 512→30 |
| Output | ID Embedding | Exp Embedding |
| | **Classification Layer** | **Classification Layer** |
| Linear | 100→ $n_{id}$ | 30→ $n_{exp}$ |
| Output | ID class | Exp class |

Table 2. The architecture of decoder

| | ID decoder | Exp decoder |
|---|---|---|
| Input | ID embedding | Exp embedding |
| Linear | ID embedding → 512 | Exp embedding → 512 |
| Activation | LeakyReLU(0.2) | |
| Linear | 512→1024 | 512→1024 |
| Activation | LeakyReLU(0.2) | |
| Linear | 1024→ $num\_verts \times 3$ | 1024→ $num\_verts \times 3$ |
| Output | $num\_verts \times 3$ | $num\_verts \times 3$ |

Table 3. The architecture of ID generator $G_{id}$, Exp generator $G_{exp}$

| | $G_{id}$ | $G_{exp}$ |
|---|---|---|
| Input | $z_{id} + z_{noise}$ | $z_{id} + z_{exp} + z_{noise}$ |
| Linear | (20 + 5)→ 512 | (20 + 20 + 5)→512 |
| Activation | LeakyReLU(0.2) | |
| Linear | 512→512 | 512→512 |
| Activation | LeakyReLU(0.2) | |
| Linear | 512→1024 | 512→1024 |
| Activation | LeakyReLU(0.2) | |
| Linear | 1024→100 | 1024→30 |
| Output | ID embedding | Exp embedding |

Table 4. The architecture of discriminator $D$

| | $D$ |
|---|---|
| Input | ID embedding + Exp embedding |
| | **Common branch** |
| Linear | (100 + 30)$\rightarrow$ 1024 |
| Activation | LeakyReLU(0.2) |
| Linear | 1024$\rightarrow$512 |
| Activation | LeakyReLU(0.2) |
| Linear | 512$\rightarrow$256 |
| Activation | LeakyReLU(0.2) |
| | **Discriminator branch** |
| Linear | 256$\rightarrow$1 |
| Output | Real or Fake |
| | **ID branch** |
| Linear | 256$\rightarrow$ $n_{id}$ |
| Output | ID class |
| | **Exp branch** |
| Linear | 256$\rightarrow$ $n_{exp}$ |
| Output | Exp class |