Supplementary material for Cross-modal Latent Space Alignment for Image to Avatar Translation

Manuel Ladron de Guevara
Carnegie Mellon University
rldg.manuel@gmail.com

Jose Echevarria
Adobe Research
echevarr@adobe.com

Yijun Li
Adobe Research
yijli@adobe.com

Yannick Hold-Geoffroy
Adobe Research
holdgeof@adobe.com

Cameron Smith
Adobe Research
casmith@adobe.com

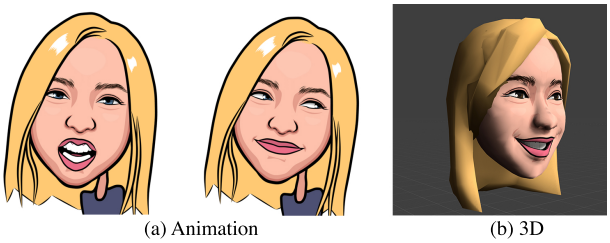
Daichi Ito
Adobe Research
dito@adobe.com

1. Applications

This paper does not delve into the various applications of parametric avatars. Nonetheless, we present the ease with which one can export avatar parameters into different environments. Compared to image-based avatars, parametric avatars offer a wider range of potential applications. Vector-based graphics, in particular, possess the advantage of being more readily exportable to other software programs for animation, 3D modeling, or virtual reality. This attribute is especially significant for game developers and animators who frequently transfer graphics between different software programs. An example of such applications is shown in Figure 1.

Animation. The first two images are snapshots of the avatar produced with our method being animated in an animation software. The third image shows the same avatar in a 3D environment. Our method allows for a much more streamlined and efficient workflow than pixel-based methods, as the avatars can be modified and used in various different contexts without having to create new images from scratch.

2D to 3D. Generating a 3D counterpart from our parametric avatar is simpler than reconstructing it from pixel-based methods. Initially, we created a template low-poly face geometry that included the head, ears, glasses, and several hair parts. This geometry was designed such that each control point from our generated parameters had a corresponding point of polygons. We also moved neighboring points accordingly, such as adjusting the head's size by moving all points of the head according to the outline of the face. Certain parameters in our avatars, such as shadows, determined the depth of the eyes. Concerning the face texture, we adapted the 2D caricature/avatar generator to render the front face and some additional parts. Then, we projected it onto the face polysurface.



(a) Animation (b) 3D

Figure 1. Applications of our system beyond images. (a) We can export our parametric vector to animation software, and (b) 3D modeling software

2. Rendering Engine

To facilitate reproducibility, we provide a detailed explanation of our rendering engine, which serves as the intermediary between the avatar parameter values and their corresponding pixel values. Each parameter value (e.g. the x, y coordinates of the control points defining the upper curve of the eyelid) is bounded within a minimum and a maximum value determined based on the relative position in the canvas. For instance, no parameter that defines an eye can be any lower than the middle height of the face. This constraints are predetermined as part of the avatar design.

Our rendering engine works by drawing each component in layers and in a specific order. Specifically, we take the vector of parameters that correspond to the face shape, denoted as $\overline{y}_{facial_shape}$, and draw a set of curves that delineate the facial outline. We then apply the base R,G,B color for the face. Next, we update the canvas to add other facial components such as the mouth, nose and eyes, as shown in Figure 2.

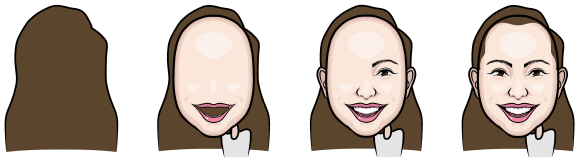


Figure 2. Rendering composition used in our vector-graphics engine. Facial parts are composited on the canvas layer by layer.

Our rendering engine, implemented in Python, is relatively straightforward and utilizes the ImageDraw module in the PIL library, which furnishes simple 2D graphics. In particular, we use the polygon function to draw closed shapes and compute the cubic Bezier equation to produce smooth open curves. Additionally, we taper the endpoints of curves to achieve a natural stroke effect. Finally, we fill the shapes with the corresponding color and maintain a black stroke for outlines.

3. High Resolution Samples

We append additional samples at a higher resolution at the end of this document to further demonstrate the capabilities of our model. Notably, we observe that our model is able to generate avatars of children despite our parametric avatar dataset not containing any children, as evidenced in Figure 3. It is worth noting that while the image encoder E_s has been trained on a dataset that includes images of children, specifically the FFHQ dataset [5], the parameter encoder and decoder E_t and D_t, respectively, have not been trained on avatar vectors corresponding to child faces. Despite this, our model is able to accurately generate avatars of children, which is a testament to the effectiveness of our parametric representation and cross-modal framework.

4. Dataset

Our original dataset contains 393 pairs of portraits and parameters made by our artist. We have a test set of 30 pairs of portraits and parameters. In the following paragraphs, we first provide more details about the vector of parameters that define an avatar, and then explain how we make artificial transformations in the training set to get up to 9970 paired samples.

4.1. Parametric Avatar

Our parametrization is designed to represent a wide range of faces from different demographics with the mininum number of parameters. Thus, we group parameters based on facial components such as $\bar{y} := \{\bar{y}_{eyes}, \bar{y}_{nose}, \bar{y}_{mouth}, ...\}$, as explained in main paper, and as shown in Table 2, for a total of 629 values. All parameters follow a normal distribution, and before feeding parameters to the model, we normalize all parameters to be in the 0-1 range, and zero out the parameters corresponding to hair, glasses and cap, as they are retrieved from a databank. The majority of parameters define x, y coordinates of control points for cubic Beizer curves and polygons, R,G,B color values, and line thickness. The use of shadows and highlights make the avatar face have more or less depth, which is found useful to represent different ethnicity.

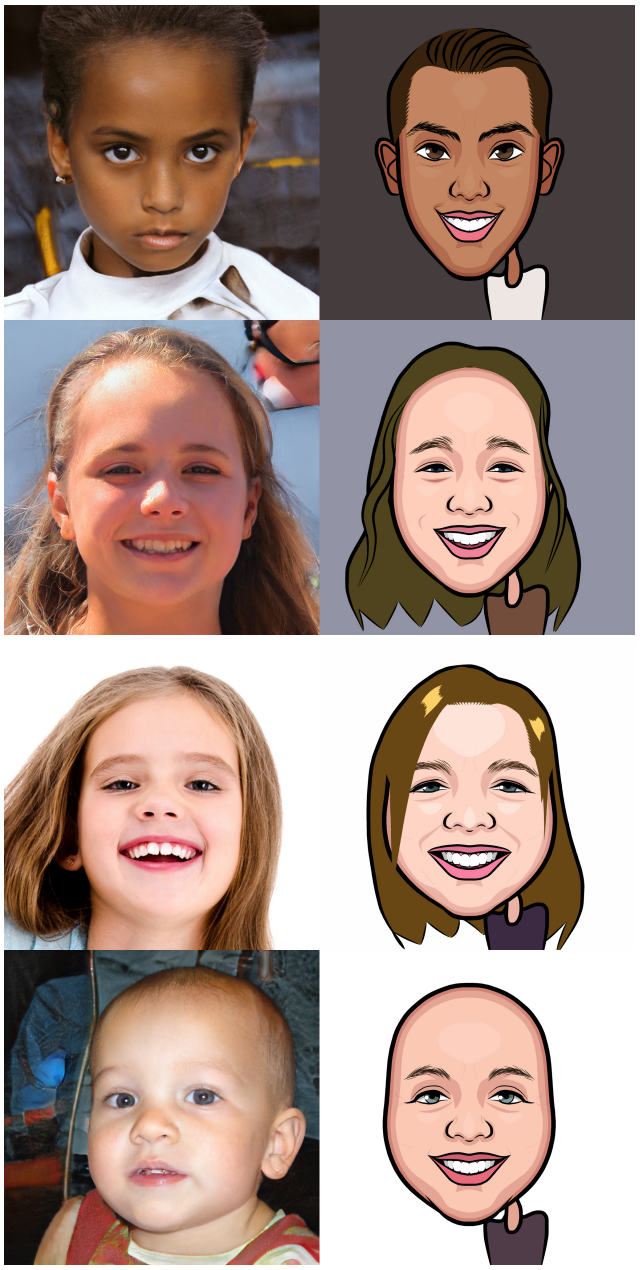


Figure 3. Examples of avatars of children. Our model generalizes to kids, even though our paired image-parameteric avatar dataset does not contain any children.

Non-trivial facial components. Some other facial parts, however, have more characteristics and variation than outlines and color, such as eyebrows. Besides the basic line shape, eyebrows present different hairline angle, volume, width, curviness, etc. A summary of parameters that define the eyebrows is shown in Table 1.

4.2. Paired Dataset

Obtaining synthetic paired dataset requires performing the same augmentation in both image and avatar modalities. We generate a total of 9577 augmentations from our original paired dataset with two different methods:

Distortion-based Augmentations We first make augmentations for the existing images by slightly distorting facial features like eyes, head shape, nose, etc. We use Photoshop's Face Aware Liquify tool for this process. A script bridges the results of this tool with the avatar parameters to create the same deformation in the corresponding paired vector of parameters. We achieve a total of 3995 paired datapoints. We use this dataset to train our model and to compare with previous methods that we present in the main paper.

Interpolations We use a software that makes in-between face photographs given two anchor photographs, A and B. We can control the interpolation amount of A and B we want to maintain, for example 20% of A and 80% of B. In order to have the same interpolation in parameter space, we make a script that interpolates between two vector of parameters to have matching image-parameter vector pairs. Note that we do not interpolate hair parameters using this method because the hair parameters do not ensure smooth interpolations. This is due to the fact that some of the hair parameters are categorical variables. For instance, hair length is a categorical value with class from 0 to 4, and hair bangs has a binary variable which turn them on and off. In total we make 5975 interpolations.

4.3. Unpaired Dataset

We can create a large amount of avatar parameters by swapping different facial components from random avatars. We synthetically create 100000 avatar vectors that do not have a corresponding portrait photograph. For our image encoder E_s, we use FFHHQ [5] (70000 aligned faces) and our augmented face dataset (9970 aligned faces) to have a total of 79970 aligned portraits.

5. Model

5.1. Model Architecture

The network architecture is shown in Figure 4. Encoders E_t, D_t and mapping network F are composed by a 2-layer MLP. The first layer is composed by a fully connected layer followed by BatchNorm and ReLU, and the last layer is a fully connected layer. These two networks output a 512-D latent vector. D_t uses a Sigmoid function after the last fully connected layer to output a 629-D vector of parameters in the range 0-1. The image encoder E_s outputs a 512-D latent vector, and the image decoder D_s takes in a 512-D latent vector and generate an image in 256x256 resolution.

5.2. Training

Table 3 shows the hyperparameters used during training in both stages, the unpaired and the paired stages. We first train each modality-specific autoencoder (image and avatar vector) independently on parallel, on large-scale unpaired data. Once the networks converge, we discard D_s, fix E_s, E_t and D_t's weights, and train the new mapping network F. Note that F is trained on paired data only.

5.3. Inference and Hair Pipeline

At inference we only have a portrait image I, and we are interested in generating a vector of parameters that define an avatar for this image. For this, we only use our image encoder E_s, our mapping network F, and our parameter decoder D_t.

Our hair pipeline leverages a pretrained network $\mathcal{H}(I) \rightarrow \hat{a}$ that takes a real photograph as input and outputs predictions of 35 attributes of the input photograph. The list of attributes is:

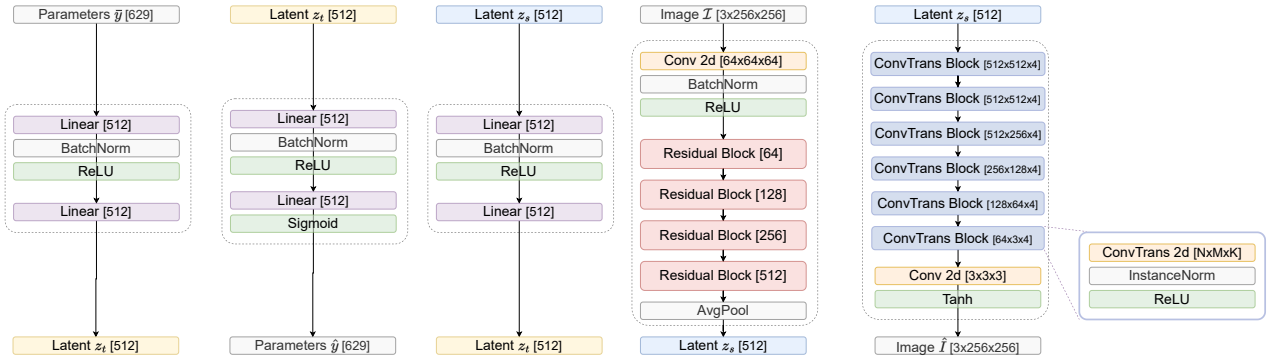
age, male, smile, beard, moustache, sideburns, facialhair, no glasses, reading glasses,sun glasses, swimming goggles,roll, yaw, pitch, anger, contempt, disgust, fear, happiness, neutral, sadness, surprise, bald, hair invisible, white hair, gray hair, blond hair, brown hair, red hair, black hair, eye makeup, lip makeup, glasses, headwear, mask.

That is, $\hat{a}$ is a 35-dimensional vector of logits. We precompute a set of vectors $\hat{a}_{i=0}^{N}$ corresponding to a bank of 400 hairstyles and accessories that are paired with photographs. As shown in Figure 4 in the main paper, at inference, and given an image I_i, we first perform a forward pass through our model, that is, $\hat{y}_i = D_t(F(E_s(I_i)))$ to get the predicted avatar parameters. We then perform a forward pass through our attribute predictor $\mathcal{H}(I_i)$ to get a query vector $\hat{a}_i$. We perform K-nearest neighbors to retrieve the most similar $\hat{a}_j$ from our databank. Because these faces are paired with ground truth avatar parameters, we can get the parameters corresponding to hair and accessories $(\hat{y}_j^{hair}, \hat{y}_j^{accs})$, respectively. We mask out the hair and accessory parameters from the predicted parameters $\hat{y}_i$ and paste the retrieved hair and accessory parameters from the hair pipeline.

Background. We can match the average background color by computing a segmentation mask I_s of the input im-

Parameter	Definition
Eyebrow volume	Density of hair lines
Initial angle	Angle of hair lines on center of the temple
Angle change	Initial angle of hair varies along the eyebrow angle gradually.
Eyebrow curve	Indicates the curviness of the eyebrow
Eyebrow hair width	Width of each hair line
Eyebrow hair length	Length of each hair line
Hair length Random	Length of hair line can be randomized. This parameter indicates random seed
Eyebrow hair spread	The root of each hair line will be perfectly aligned if this is zero. If bigger than zero, they will be randomly scattered.
Hair spread	Control spread of hair on eyebrows.
Eyebrow hair spread Y	Scatters hairlines only in Y direction to make the eyebrow taller
Fewer hair at end	Controls the number of the hair line at the edge of eyebrow
Hair spread at end	Controls taper at the edge of eyebrow
Eyebrow coverage	Scatters hairlines locally in the vertical direction

Table 1. Breakdown of the parameters that define the eyebrows.



(a) Parameter Encoder E_t (b) Parameter Decoder D_t (c) Mapping Network F (d) Image Encoder E_s (e) Image Decoder D_s (f) ConvTrans Block

Figure 4. Model architecture of the different components of our model.

age I. We use a BiseNet [13]-based network [8] to get I_s. We then calculate the mean RGB value in I's background to feed it to the vector graphics engine.

6. Ablation Studies

6.1. Pretrained Image Encoders Ablations

We are interested in testing the causal relationship between our face encoder and identity preservation capabilities of our model. That is, how well our encoder E_s helps preserve identity, keeping the rest of the networks (F and D_t) fixed. We generate avatars using off-the-shelf pretrained face encoders and compare these encoders with ours. ArcFace [2] is trained on a much larger dataset (1M images), and uses a ResNet100 [3] as a backbone. FaceNet [12] uses an inception model as backbone and is trained on 8M identities. pSp's encoder [11] uses a feature pyramid network [7] to get the representations of the input images, and it is trained on FFHQ dataset [5]. All encoders generate a latent vector $w \in \mathbb{R}^{512}$, except for pSp which projects to $\mathcal{W}+$. Here, we define w to be the average vector.

Figure 5 shows comparisons of different encoders. Overall, our encoder and pSp encoder output more successful results in terms of identity preservation. ArcFace struggles preserving the identity of the input more than the rest of the encoders, and FaceNet (column (c)) preserves the overall identity but fails to capture finer details in the mouth: in the first and second rows, it wrongly leaves a gap between the upper teeth and the bottom lip, and in the fourth row, it wrongly omits this gap. When the network uses our encoder, the shape of the mouth is generally better than other decoders. Altogether, our pipeline design allows for other pretrained encoders to be plugged-in without significantly degrading the identity preservation of the generated avatars.

Avatar part	# parameters	Method
Facial shape	56	Learned
Nose	33	Learned
Eyebrows	25	Learned
Eyes	83	Learned
Mouth	54	Learned
Ears	12	Learned
Highlights	80	Learned
Moustache	20	Learned
Beard	22	Learned
Hair	152	KNN Retrieval
Glasses	41	KNN Retrieval
Cap	46	KNN Retrieval
Body	5	Learned

Table 2. Avatar parameterization by facial components. Hair, glasses and cap parameters are not learned by our main model. Instead, they are retrieved by a separate pipeline which we refer to as "hair pipeline" for brevity.

	Unpaired training	Paired training
Epochs	20	10
Batch size	24	48
Optimizer	Adam	Adam
Learning Rate	0.0002	0.0002
Scheduler	None	Plateau
Latent Space Dim.	512	512
Optimizing Nets	E_t, D_t, E_s, D_s	F

Table 3. Training Hyper-parameters

6.2. Effect of Amount of Data

We are interested in testing the amount of paired data that is needed so that our model does not break. Figure 6 shows examples of our model trained on different amounts of data. Specifically, we train our model on 9970 image-avatar parameters pairs (column (b)), 3995 (column (c)), 2000 (column (d)), 1000 (column (e)), and 500 (column (f)). As noted in the main paper, our model starts breaking identity preservation when trained on 2000 paired samples or fewer, as shown in columns d, e, and f. When trained on 1000 samples or fewer, the model produces exaggerated facial features, such as a bigger and narrower mouth (third and last rows), larger head size (first, second, and fourth rows), and nose misplacement (second, third, and last row). When trained on nearly 4000 samples (column (c)), our model maintains the input's identity overall. However, it sometimes misses more nuanced features, making errors in scale and placement. When trained on nearly 10000 paired samples, our model captures much better identity features. There is a visual relation of amount of data and avatar quality, and our method needs at least 2000 paired data samples in order to preserve identity and generate high quality avatars.

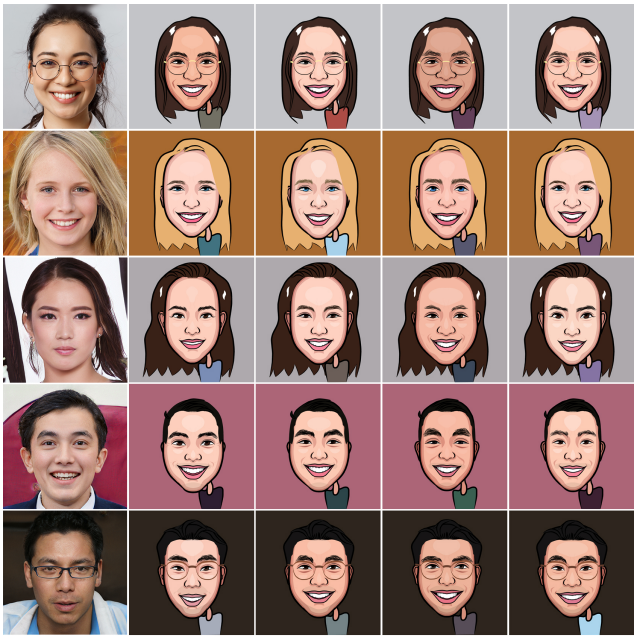


Figure 5. Avatars generated by our method using different pre-trained encoders.

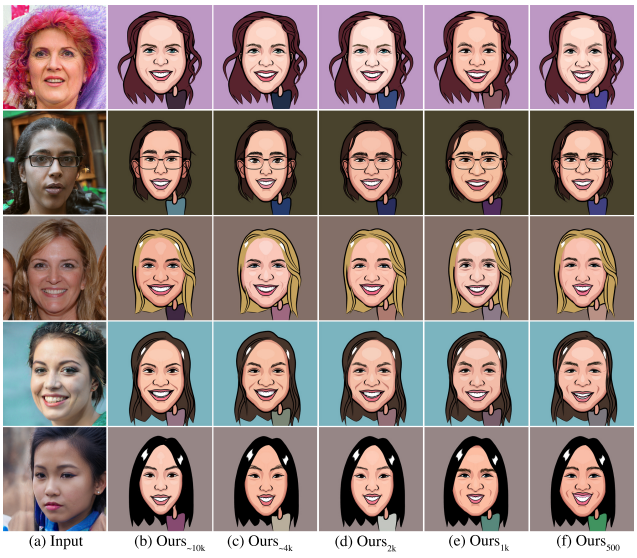


Figure 6. Effect of amount of paired data. (b) Our model trained on 9970 pairs. (c) Our model trained on 3995 pairs. (d) Our model trained on 2000 pairs. (e) Our model trained on 1000 pairs. (f) Our model trained on 500 pairs

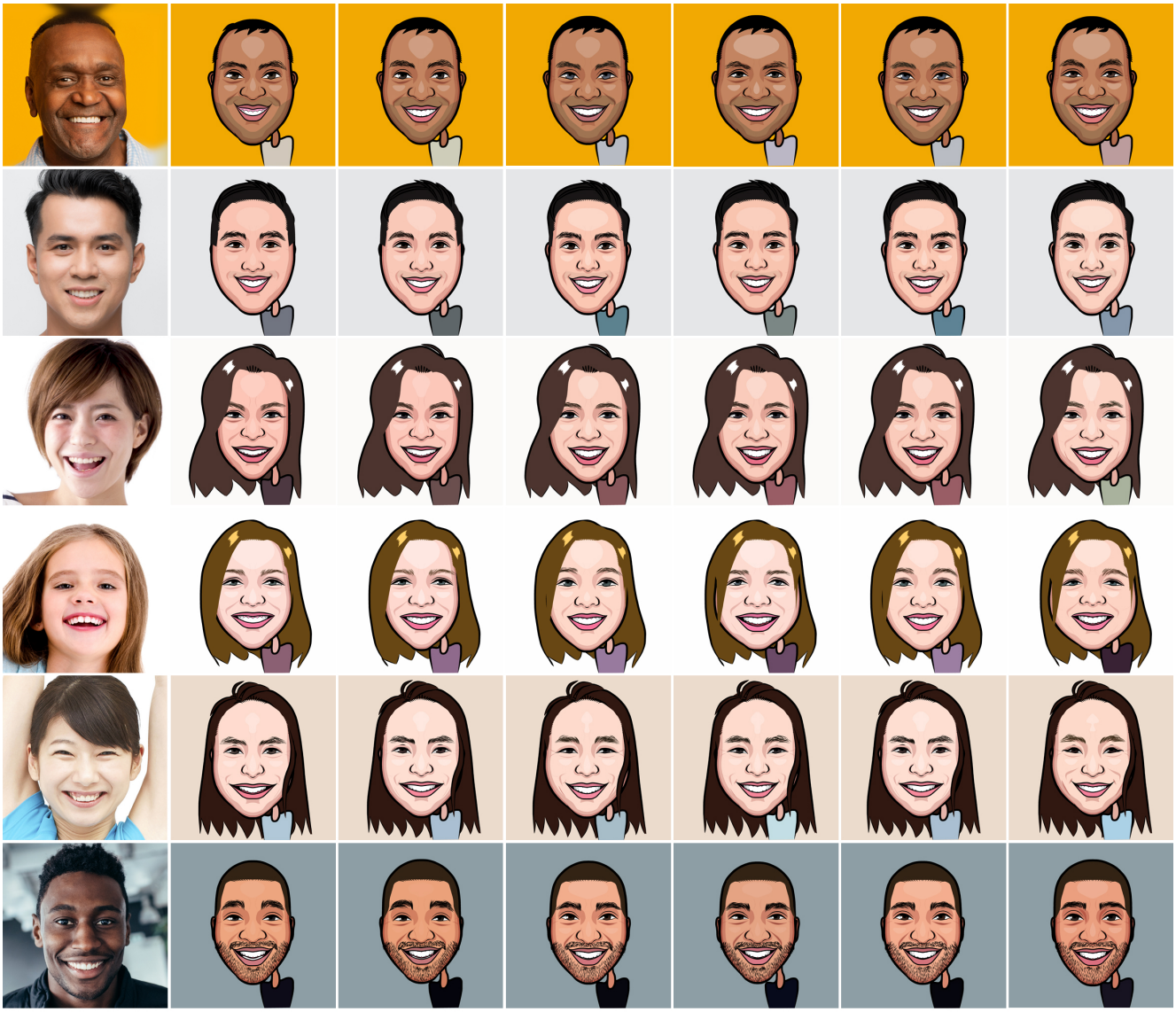


Figure 7. Model variations without proposed losses. (b) only uses reconstruction loss $\mathcal{L}_{rec}$ between generated and ground truth parameters. (c) only uses weight alignment loss $\mathcal{L}_w$. (d) uses reconstruction $\mathcal{L}_{rec}$ and cross modal loss $\mathcal{L}_{cm}$ but does not use weight alignment loss $\mathcal{L}_w$ (e) uses all except for the MSE loss term in $\mathcal{L}_{cm}$. (f) uses all except for the cosine similarity term in $\mathcal{L}_{cm}$ (g) uses all losses

6.3. Loss Ablations

We provide more visual comparisons between our model using different losses in Figure 7. This ablation study tests our mapping function F under different objectives. For an easier evaluation, we re-state our objective functions here. Our full objective is composed by a (1) reconstruction loss $\mathcal{L}_{rec} = (\hat{y} - \bar{y})^2$, where $\hat{y}$ is the predicted parameters output by a forward pass through image encoder, mapping network and parameter decoder $D_t(F(E_s(I)))$ given an image I, and $\bar{y}$ is the vector of parameters that is paired to I; and by a (2) alignment loss. Our alignment loss is composed by three terms: $\mathcal{L}_{mse}$, $\mathcal{L}_{cossim}$, and a weight regularization term $\mathcal{L}_w$. The first two measure a predicted latent code given by a forward pass through the image encoder and mapping network $z_f = F(E_s(I))$ and the output of a pretrained parameter encoder E_t that we refer as the "expert" network, $z_t = E_t(\bar{y})$. Our weight regularization term imposes a regularization on the last layer of F's weights, built on the assumption that F's last layer should be as similar as E_t's last layer weights, since the former wants to project a vector into the same la-

tent space as the latter, and both networks are designed to be a 2-layer MLP.

All columns in Figure 7 keep the reconstruction loss $\mathcal{L}_{rec}$. Column (b) removes the alignment module entirely and only keeps the reconstruction loss. The rest of the columns (c,d,e,f, and g) keep different terms from the alignment module. Specifically, column (c) only maintains the weight regularization term $\mathcal{L}_w$. Column (d) does the opposite and only keeps this term (besides reconstruction loss). Columns (e) and (f) remove $\mathcal{L}_{mse}$, and $\mathcal{L}_{cossim}$ from our alignment module, respectively. Last column (g) keeps all the losses.

Without our alignment module (column (b)), the model does not capture well the identity of the input person. From here, the rest of the columns do a relatively good job at preserving identity. However, by just using our weight regularization (column (c)), the model fails to capture some facial components that contribute to the entire identity. For instance, in the third row, the eyes and the nose are closer together than the input image. In the fourth row, the model is generating somehow features of an older face. In the fifth row, the generated eyes do not register the smiling and consequently more stretched eyes from the input image. In the bottom row, the eyes are not as big as the person in the input image.

Without our weight regularization term (column (d)), the model does a better job at fixing the problems explained in the previous paragraph, but still generates facial features that are a bit off from the input image. For example, in the bottom row, the nose shape is different from the input image. In the fifth row, the cheeks' wrinkles are not as pronounced as in the input image. In the fourth row, the eyes of the little girl are not accurate, and a similar inaccuracy happens in the first row.

Without the MSE term in our alignment loss (column (e)), the model fails to register the eye shape accurately in the last two rows, as well as the second row. In the first row, the face shape or outline is rounder than the input image. In the second row, the nose is not as accurate as the one generated using all loses (column (g)). However, it seems like this loss term might be the less important.

Without the cosine similarity loss term (column (f)), the model generates less accurate avatars than column (e) and (g). In the first row, the eyes are bigger and have a different color than the input image. In the third row, the face shape is longer and less round than the input image. In the fourth row the eyes do not look similar to the input image, and the mouth does not register the open gesture (see how columns (e) and (g) do present a gap between the upper teeth and the bottom lip). The identity of the last two rows is not successfully preserved.

By using all proposed losses our model consistently preserve identity better across age, gender and ethnicity.

Method	Gen.	Inv.	Hair	Total
U-GAT-IT [6]	0.0108			0.0108
GAN-Adapt. [10]	0.013	60.975		60.988
StarGAN-v2 [1]	1.222			1.222
MSGAN-pix2pix [9]	0.0102			**0.0102**
StyleCariGAN [4]	0.081	77.946		78.027
Ours	1.229*		2.747	3.976

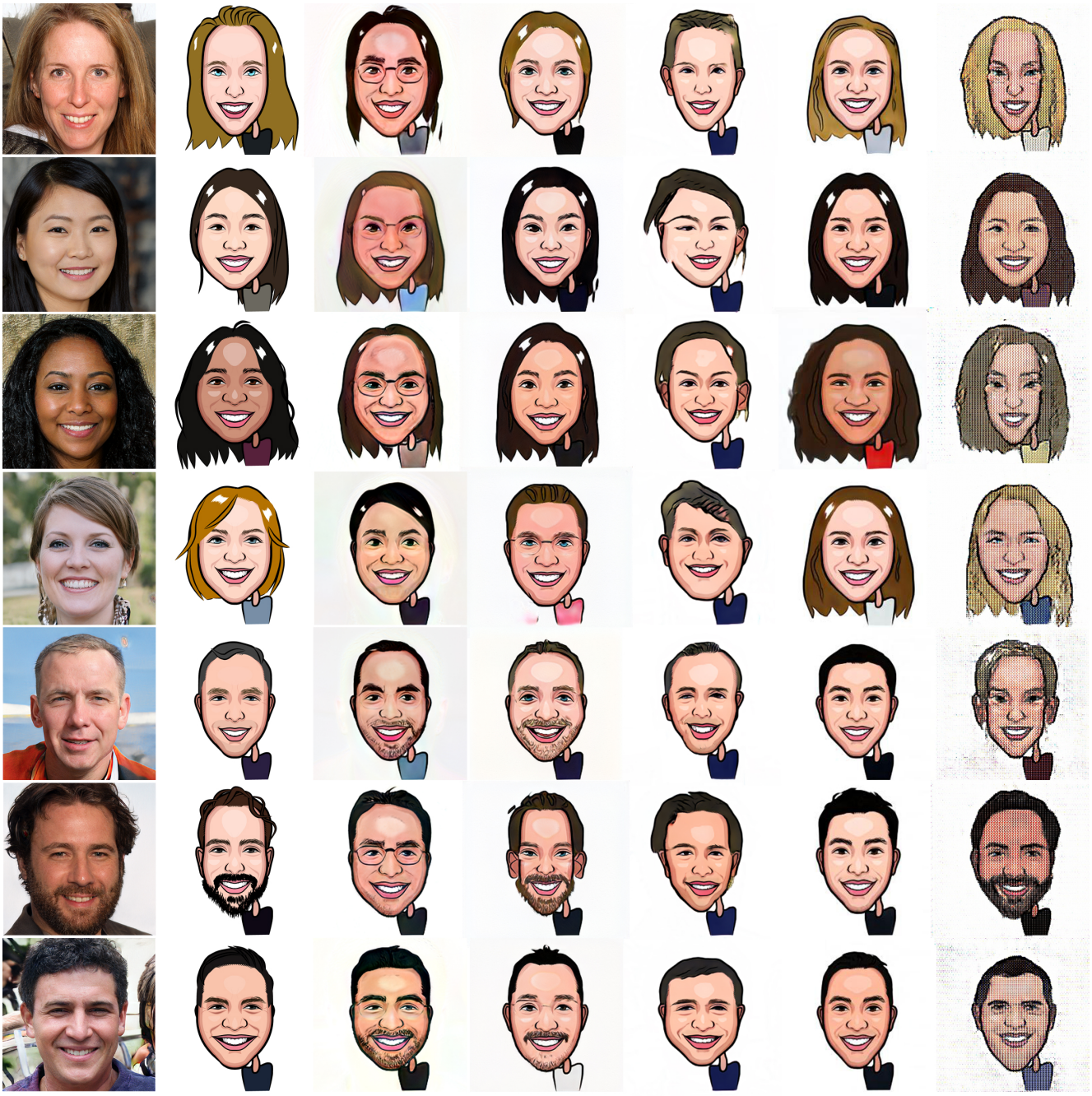
Table 4. Inference Runtime computed in seconds. GAN-Inversion is needed in StyleGAN2-based methods, where the latent vector corresponding to the input image needs to be found. Inversion is computed over 1000 iterations. *Our forward includes the forward pass through the inference module and the time that of the vector graphics engine to render the avatar in image space.

7. Comparison with Previous Methods

Figure 8 shows more examples of comparisons with previous methods. We include MSGAN-Pix2pix [9] in this figure. We use official implementations of each method and train with the default hyperparameters. Overall, we see our method generates better quality and better identity preserved avatars. The GAN-Adaptation method [10] (column (b)) works better for males than females. Hair color is generally not well preserved (all rows except for the last two), and it wrongly adds eyeglasses on some inputs (first three rows and sixth). It generally preserves well cheek wrinkles. StyleCariGAN [4] (column (d)) generates diverse and good quality avatars, but sometimes it struggles preserving the identity of the input. In the top row the avatar does not register the elongated facial shape of the input. In the second row, the avatar face is not as round as in the photograph. In the fourth row the model is generating eyeglasses, and in the fifth and sixth row, identity is almost completely missing. U-GAT-IT [6] (column (e)) produces smooth colors and generally preserves identity better than other previous methods. However, the generated avatars present some deformations that degrade the quality of the avatar. It struggles with long hair, but in general it preserves well key facial features such as eyebrows, eyes, nose or face shape. StarGANv2 [1] (column (f)) produces higher quality avatars than other previous methods, but identity is not as well preserved. The last three rows have similar faces. The first row does not preserve the elongated face from the input. MSGAN-Pix2pix [9] (column (g)) generates diverse avatars and generally registers well different hairstyles and preserves relatively well facial shape (except for the example in the third row). The quality of the image generated by this method is compromised by the artifacts the model generates.

7.1. Inference Runtime

Table 4 shows the inference runtime of our method in comparison to previous methods. Methods free of GAN-



(a) Input (b) Ours (c) GAN-Adapt. (d) StyleCariGAN (e) U-GAT-IT (f) StarGANv2 (g) MSGAN-Pix2pix

Figure 8. Comparison of our model with state-of-the-art methods. All methods are trained on the same dataset, comprising 3995 image-avatar pairs.

inversion or render engines such as U-GAT-IT [6] and MSGAN-Pix2pix [9] are the fastest to generate avatars, as these models only need a forward pass through their networks to directly generate the avatar in image space. GAN-Inversion methods such as StyleCariGAN [4] and GAN-Adaptation [10] are slower due to the optimization step needed to compute the StyleGAN latent vector that corresponds to the input image. These methods need around one minute to generate an avatar. Our method generates avatar images in around 4 seconds. This is due to the rendering step that converts the generated vector of parameters to an image ($\sim 1.2s$), and the hair and accessory retrieval

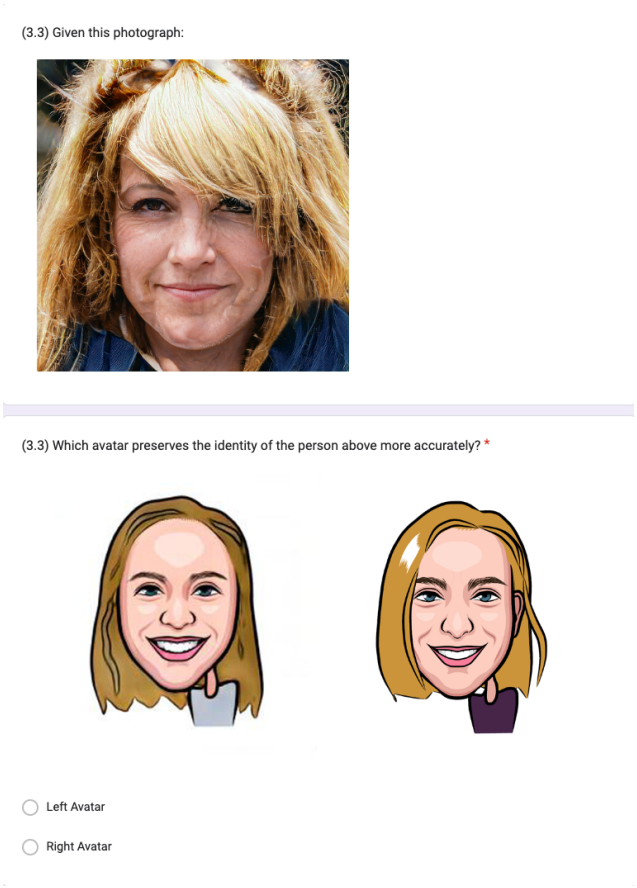


Figure 9. Example of question in our user study

pipeline ($\sim 2.7s$).

7.2. User Study

In order to ensure fairness in the user study, we provide detailed instructions that guide the selection process. We explicitly ask participants to not consider image quality when answering. That is, not to consider definition, resolution or crispness of the images. An example of a question that evaluates identity is shown in Figure 9.

References

[1] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020. 7

[2] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019. 4

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[4] Wonjong Jang, Gwangjin Ju, Yucheol Jung, Jiaolong Yang, Xin Tong, and Seungyong Lee. Stylecarigan: caricature generation via stylegan feature map modulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. 7, 8

[5] Tero Karras, Samuli Laine, and Timo Aila. 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. 2, 3, 4

[6] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. *arXiv preprint arXiv:1907.10830*, 2019. 7, 8

[7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 4

[8] Ingo Lütkebohle. Face Parsing. `https://github.com/zllrunning/face-parsing.PyTorch`, 1028. 4

[9] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1429–1437, 2019. 7, 8

[10] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A. Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10743–10752, June 2021. 7, 8

[11] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021. 4

[12] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 4

[13] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 4

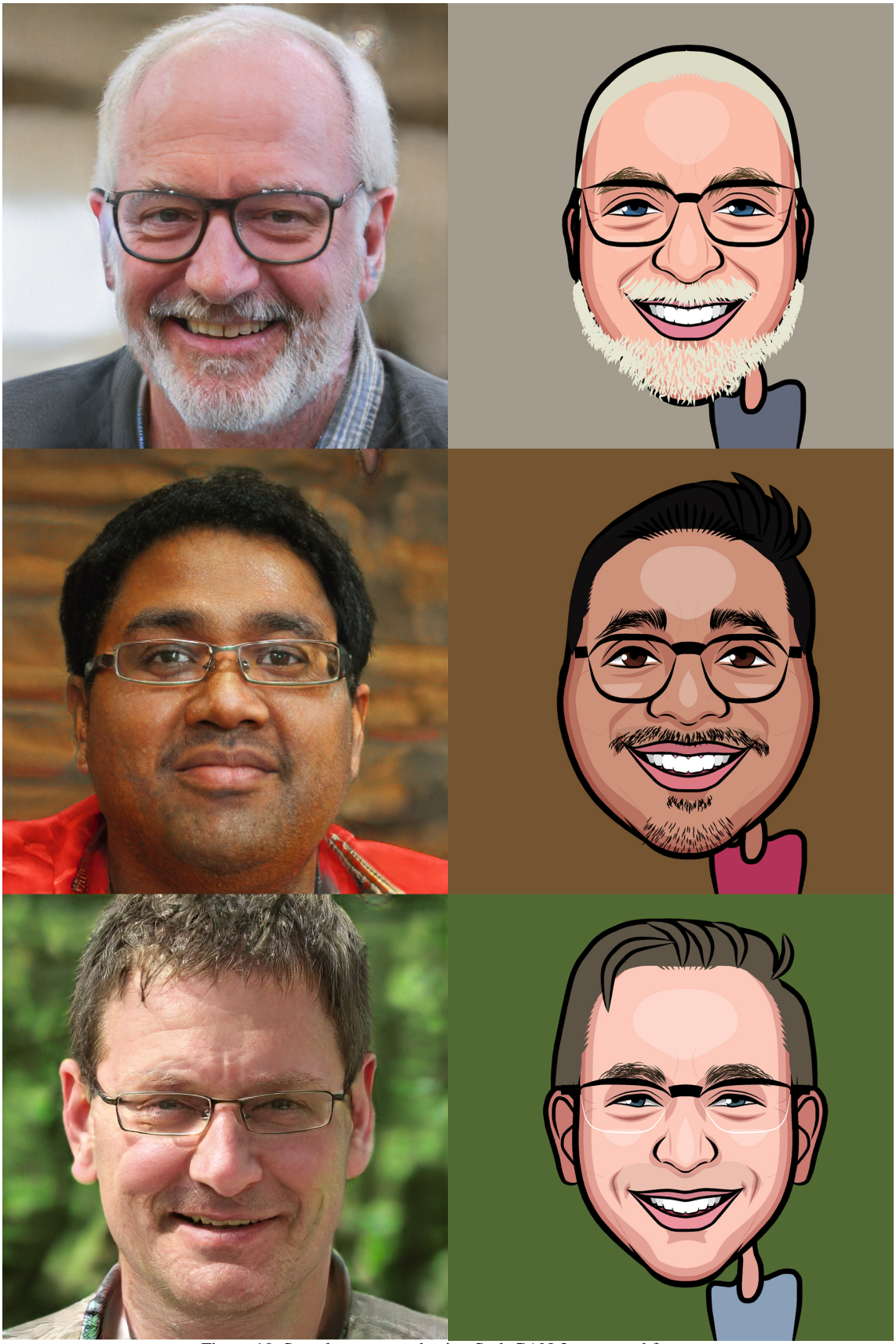


Figure 10. Samples generated using StyleGAN-2 generated faces.

Figure 11. Samples generated using StyleGAN-2 generated faces.

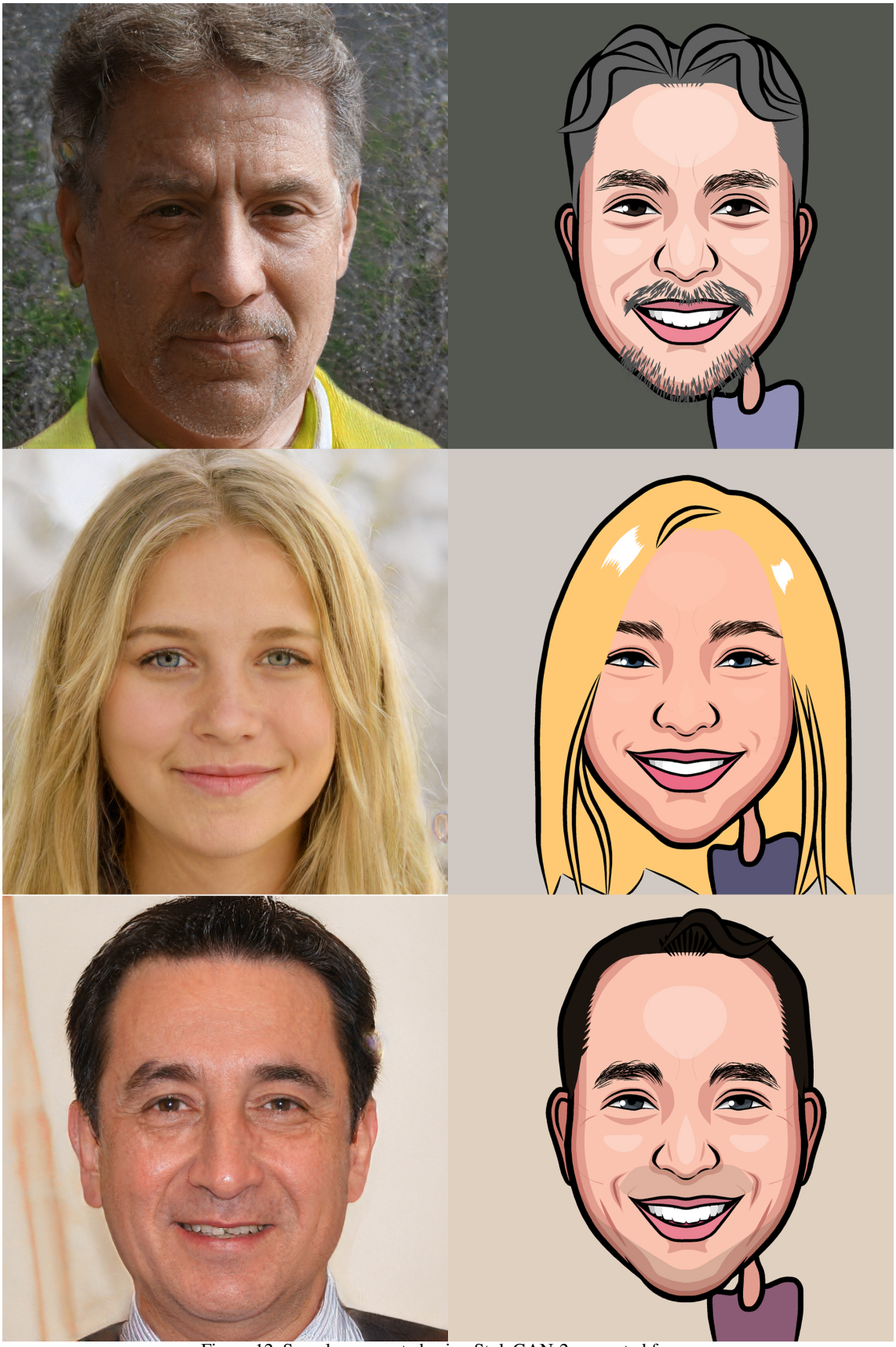


Figure 12. Samples generated using StyleGAN-2 generated faces.

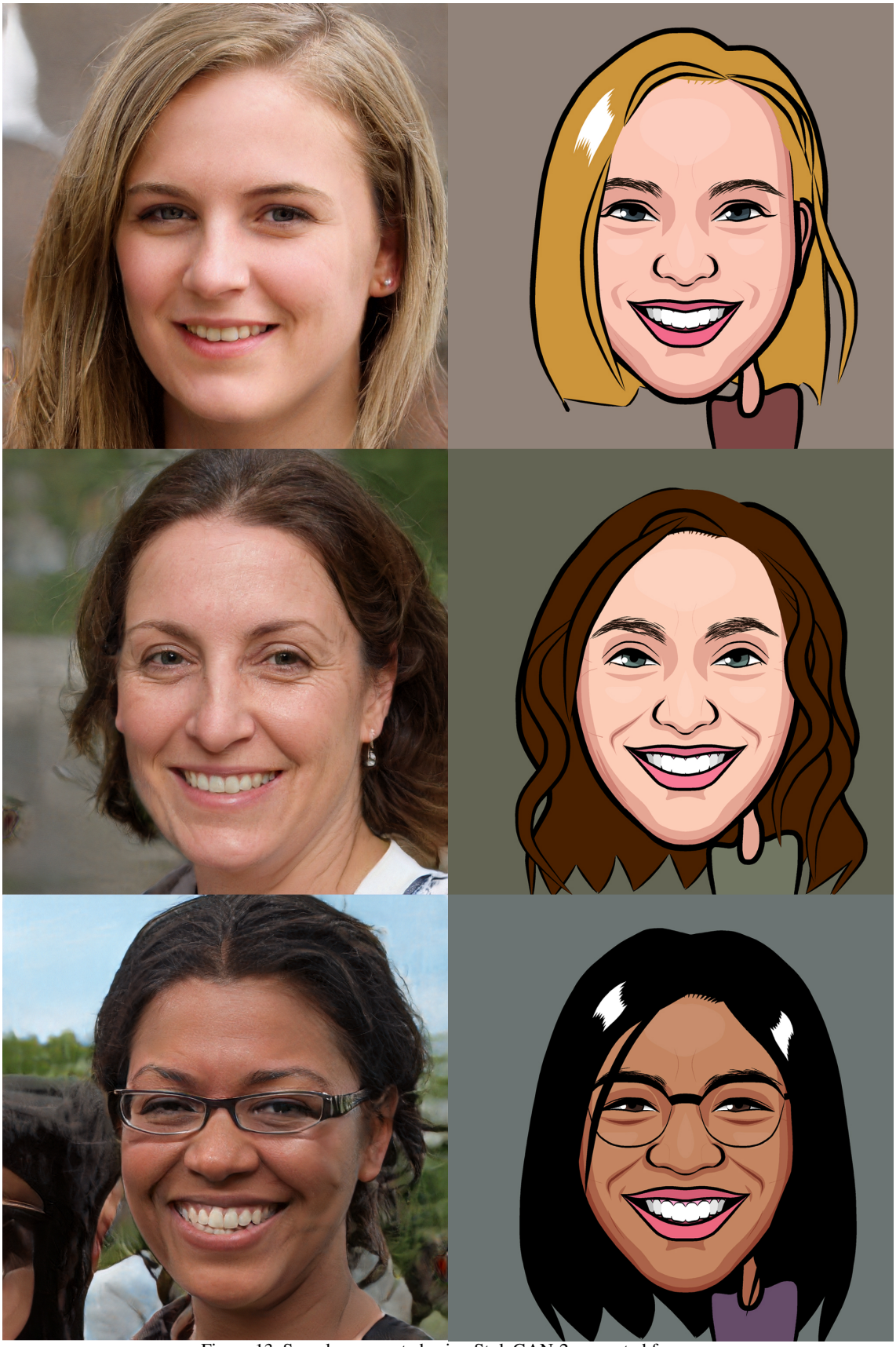


Figure 13. Samples generated using StyleGAN-2 generated faces.

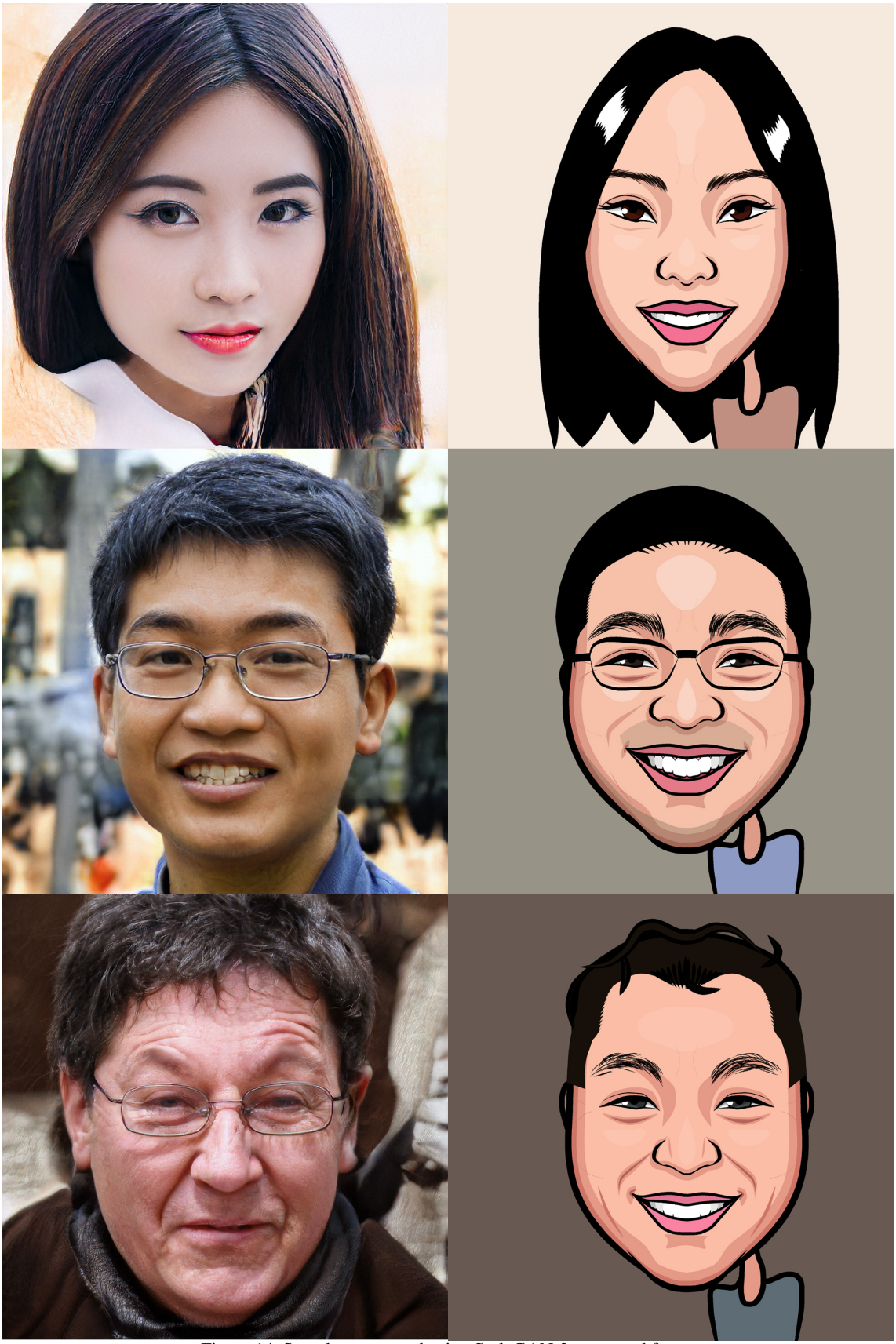


Figure 14. Samples generated using StyleGAN-2 generated faces.

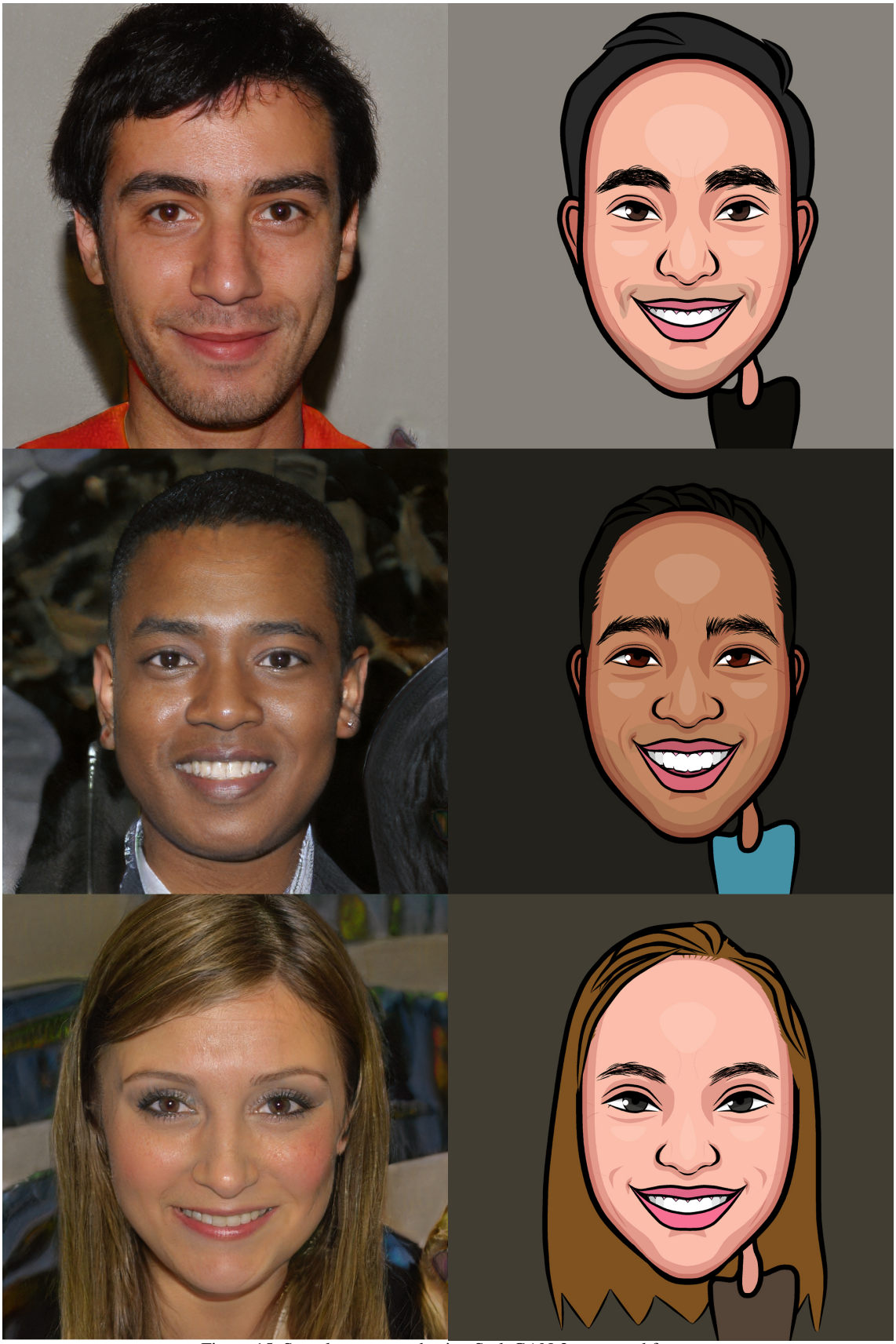


Figure 15. Samples generated using StyleGAN-2 generated faces.

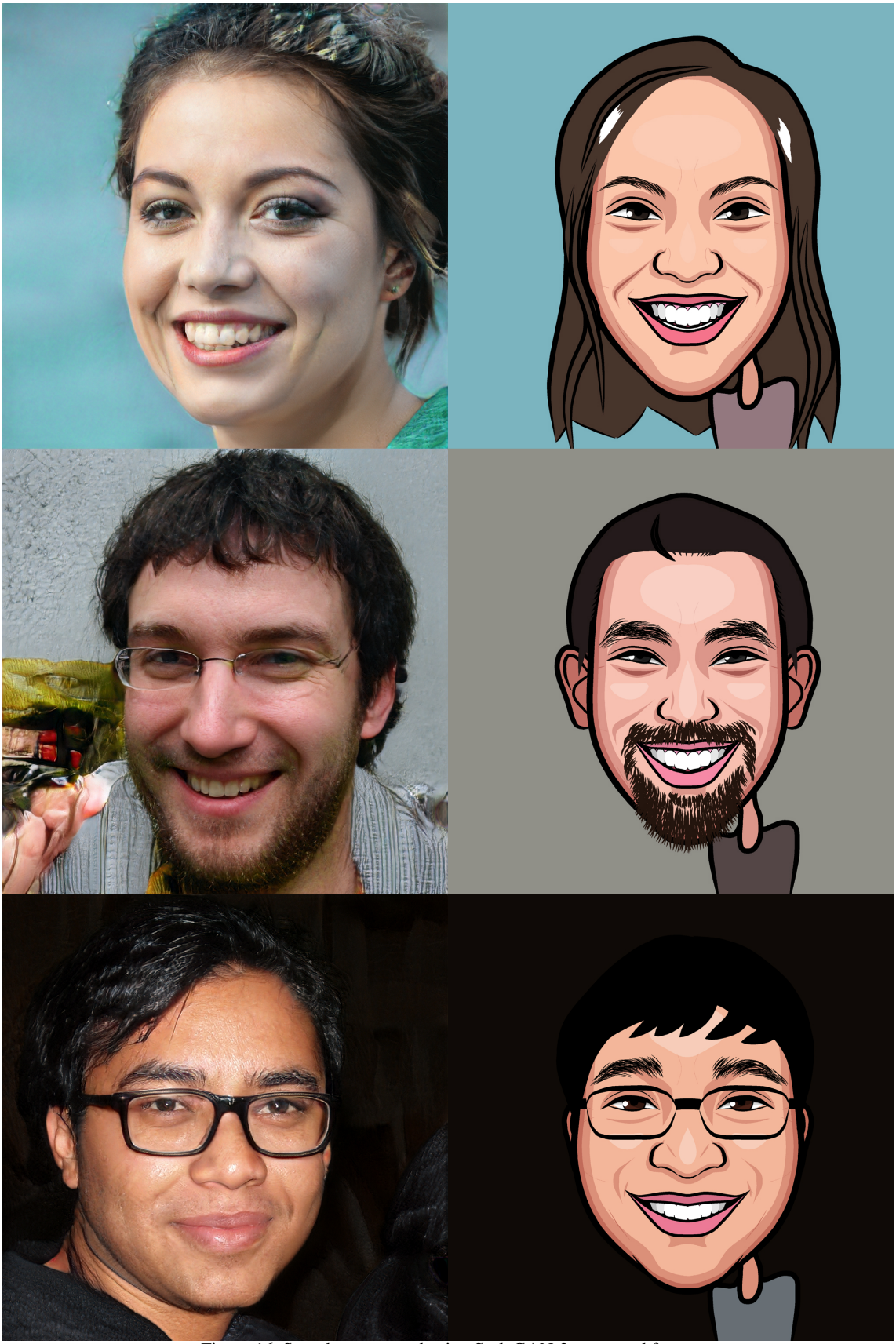


Figure 16. Samples generated using StyleGAN-2 generated faces.