AgileAvatar: Stylized 3D Avatar Creation via Cascaded Domain Bridging

Shen Sang
ByteDance
Mountain View, USA

Tiancheng Zhi
ByteDance
Mountain View, USA

Guoxian Song
ByteDance
Mountain View, USA

Minghao Liu
UC Santa Cruz
Santa Cruz, USA

Chunpong Lai
ByteDance
Mountain View, USA

Jing Liu
ByteDance
Mountain View, USA

Xiang Wen
ByteDance
Hangzhou, China

James Davis
UC Santa Cruz
Santa Cruz, USA

Linjie Luo
ByteDance
Mountain View, USA

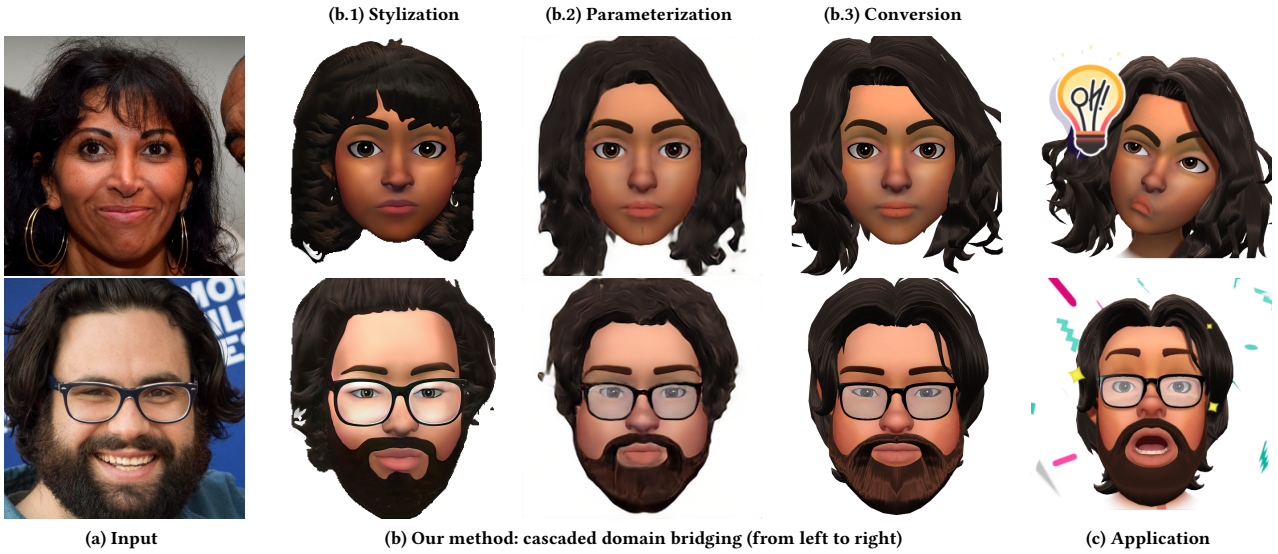


Figure 1: (a) Given a front-facing user image as input, (b) our method progressively bridges the domain gap between real faces and 3D avatars through three stages: (b.1) The stylization stage performs an image space translation to generate a stylized portrait while normalizing expressions. (b.2) The parameterization stage uses a learned model to find avatar parameters which match the results of stylization. (b.3) The conversion stage searches for a valid avatar vector matching the parameterization that can be rendered by the graphics engine. (c) The output is a user editable 3D model which can be animated and applied to various applications, for example personalized emoji. ©H JACQUOT and Montclair Film.

ABSTRACT

Stylized 3D avatars have become increasingly prominent in our modern life. Creating these avatars manually usually involves laborious selection and adjustment of continuous and discrete parameters and is time-consuming for average users. Self-supervised approaches to automatically create 3D avatars from user selfies promise high quality with little annotation cost but fall short in application to stylized avatars due to a large style domain gap. We propose a novel self-supervised learning framework to create high-quality stylized 3D avatars with a mix of continuous and discrete parameters. Our cascaded domain bridging framework first leverages a modified portrait stylization approach to translate input selfies into stylized avatar renderings as the targets for desired 3D avatars. Next, we find the best parameters of the avatars to match the stylized avatar renderings through a differentiable imitator we train to mimic the avatar graphics engine. To ensure we can effectively optimize the discrete parameters, we adopt a cascaded relaxation-and-search pipeline. We use a human preference study to evaluate how well our method preserves user identity compared to previous work as well as manual creation. Our results achieve much higher preference scores than previous work and close to those of manual creation. We also provide an ablation study to justify the design choices in our pipeline.

Authors' email addresses: Shen Sang: shen.sang@bytedance.com; Tiancheng Zhi: tiancheng.zhi@bytedance.com; Guoxian Song: guoxiansong@bytedance.com; Minghao Liu: mliu40@ucsc.edu; Chunpong Lai: chunpong.lai@bytedance.com; Jing Liu: jing.liu@bytedance.com; Xiang Wen: sidao@bytedance.com; James Davis: davis@cs.ucsc.edu; Linjie Luo: linjie.luo@bytedance.com.

CCS CONCEPTS

• **Computing methodologies** → **Non-photorealistic rendering**.

KEYWORDS

Avatar Creation, Human Stylization

1 INTRODUCTION

An attractive and animatable 3D avatar is an important entry point to the digital world that has become increasingly prominent in modern life for socialization, shopping and gaming etc. A good avatar should be both personalized (reflecting the person's unique appearance) and good-looking. Many popular avatar systems adopt cartoonized and stylized designs for their playfulness and appealingness to the users such as Zepeto[1] and ReadyPlayer[2]. However, creating an avatar manually usually involves laborious selections and adjustments from a swarm of art assets which is both time-consuming and difficult for average users with no prior experience.

In this paper, we study automatic creation of stylized 3D avatars from a single front-facing selfie image. To be specific, given a selfie image, our algorithm predicts an *avatar vector* as the complete configuration for a *graphics engine* to generate a 3D avatar and render avatar images from predefined 3D assets. The avatar vector consists of parameters specific to the predefined assets which can be either continuous (e.g. head length) or discrete (e.g. hair types).

A naive solution is to annotate a set of selfie images and train a model to predict the avatar vector via supervised learning. However, large scale annotations are needed to handle a large range of assets (usually in the hundreds). To alleviate the annotation cost, self-supervised methods [Shi et al. 2019, 2020] are proposed to train a differentiable *imitator* that mimics the renderings of the graphics engine to automatically match the rendered avatar image with the selfie image using various losses of identity and semantic segmentation. While these methods proved effective to create semi-realistic avatars close to user's identity, they fall short in application to stylized avatars since the style domain gap between selfie images and stylized avatars are too large (see Fig. 7).

Our main technical challenges are two folds: (1) the large domain gap between user selfie images and stylized avatars and (2) the complex optimization of a mix of continuous and discrete parameters in the avatar vector. To address these challenges, we formulate a cascaded framework which progressively bridge the domain gap while ensuring optimization convergence on both continuous and discrete parameters. Our novel framework consists of three stages: Portrait Stylization, Self-supervised Avatar Parameterization, and Avatar Vector Conversion. Fig. 1 shows the domain gap gradually bridged across the three stages, while the identity information (hair style, skin tone, glasses, etc.) is maintained throughout the pipeline.

First, the Portrait Stylization stage focuses on 2D real-to-stylized visual appearance domain crossing. This stage translates input selfie image to a stylized avatar rendering and remains in image space. Naively applying existing stylization methods [Pinkney and Adler 2020; Song et al. 2021] for translation will retain factors such as expression, which would unnecessarily complicate later stages of our pipeline. Thus, we create a modified variant from AgileGAN [Song et al. 2021] to ensure uniformity in expression while preserving user identity.

[1]https://zepeto.me/
[2]https://readyplayer.me/

Next, the Self-Supervised Avatar Parameterization stage focuses on crossing from image pixel domain to avatar vector domain. We observed that strictly enforcing parameter discreteness causes optimization to fail to converge. To address this, we use a *relaxed* formulation called a *relaxed avatar vector* in which discrete parameters are encoded as continuous one-hot vectors. To enable differentiability in training, we trained an imitator in similar spirit to F2P [Shi et al. 2019] to mimic the behavior of the non-differentiable engine.

Finally, the Avatar Vector Conversion stage focuses on domain crossing from the relaxed avatar vector space to the *strict avatar vector space* where all the discrete parameters are one-hot vectors. The strict avatar vector can then be used by the graphics engine to create final avatars and for rendering. We employ a novel search process that leads to better results than direct quantization.

To evaluate our results, we use a human preference study to evaluate how well our method preserves personal identity relative to baseline methods including F2P [2019] as well as manual creation. Our results achieve much higher scores than baseline methods and close to those of manual creation. We also provide an ablation study to justify the design choices in our pipeline.

In summary, our technical contributions are:

- A novel self-supervised learning framework to create high-quality stylized 3D avatars with a mix of continuous and discrete parameters;
- A novel approach to cross the large style domain gap in stylized 3D avatar creation using portrait stylization;
- A cascaded relaxation and search pipeline that solves the convergence issue in discrete avatar parameter optimization.

2 RELATED WORK

3D Face Reconstruction: Photorealistic 3D face reconstruction from images has been studied extensively for many years. Extremely high quality models can be obtained using gantries with multiple cameras followed by a stereo or photogrammetry reconstruction [Beeler et al. 2010; Yang et al. 2020]. When only a single image is available, researchers leverage a parameterized 3D morphable model to reconstruct realistic 3D faces [Blanz and Vetter 1999; Chen and Kim 2021; Deng et al. 2019b; Peng et al. 2017; Xu et al. 2020]. Excellent surveys [Egger et al. 2020; Zollhöfer et al. 2018] exist providing great insights in this direction. These methods focus on an accurate reconstruction of the real human, and the model parameters often lack physical meaning. In contrast our work focuses on cross domain creation of a stylized avatar which has parameters with direct meaning to casual users.

3D Caricature: Non-photorealistic 3D face reconstruction has also received interest recently, a popular style being caricature. Qiu et al. [2021] created a dataset of 3D caricature models for reconstructing meshes from caricature images. Some works generate caricature meshes by exaggerating or deforming real face meshes, with [Cai et al. 2021; Wu et al. 2018] or without [Lewiner et al. 2011; Vieira et al. 2013] caricature image input. Sketches can be used to guide the creation [Han et al. 2017, 2018]. Recent works [Li et al. 2021; Ye et al. 2021] use GANs to generate 3D caricatures given real images. However, these methods are designed for reconstructing caricature meshes and/or textures while we focus on cartoonish avatars constrained by parameters with semantic meaning.

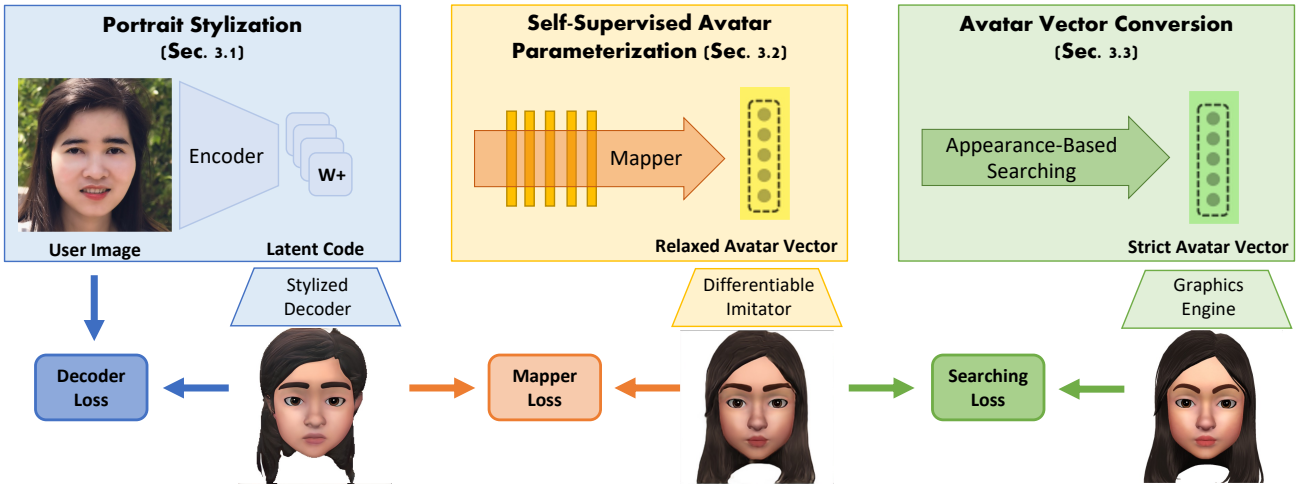


Figure 2: Pipeline. Our framework consists of three modules: Portrait Stylization for image-space real-to-stylized domain crossing, Self-supervised Avatar Parametrization for recovering relaxed avatar vector from the stylization latent code, and Avatar Vector Conversion for discretizing the predicted relaxed avatar vector into a strict avatar vector that can be taken by the graphics engine directly. ©NGÁO STUDIO.

Game Avatars: Commercial products such as Zepeto and Ready-Player use a graphics engine to render cartoon avatars from user selfies. While no detailed description of their methods exists, we suspect these commercial methods are supervised with a large amount of manual annotations, something this paper seeks to avoid.

Creating semi-realistic 3D avatars has also been explored [Cao et al. 2016; Hu et al. 2017; Ichim et al. 2015; Luo et al. 2021]. Most relevant to our framework, Shi et al. [2019] proposed an algorithm to search for the optimal avatar parameters by comparing the input image directly to the rendered avatar. Follow-up work improves efficiency [Shi et al. 2020], and seeks to use the photograph's texture to make the avatar match more closely [Lin et al. 2021]. These efforts seek to create a similar looking avatar, while this paper seeks to create a highly stylized avatar with a large domain gap.

Portrait Stylization: Many methods for non-photorealistic stylization of 2D images exist. Gatys et al. [2016] proposed neural style transfer, matching features at different levels of CNNs. Image-to-image models focus on the translation of images from a source to target domain, either with paired data supervision [Isola et al. 2017] or without [Park et al. 2020; Zhu et al. 2017]. Recent development in GAN inversion [Richardson et al. 2021; Tov et al. 2021] and interpolation [Pinkney and Adler 2020] methods make it possible to achieve high quality cross-domain stylization [Cao et al. 2018; Song et al. 2021; Zhu et al. 2021]. The end result of these methods are in 2D pixels space and directly inspire the first stage of our pipeline.

3 PROPOSED APPROACH

Our cascaded avatar creation framework consists of three stages: Portrait Stylization (Sec. 3.1), Self-supervised Avatar Parameterization (Sec. 3.2), and Avatar Vector Conversion (Sec. 3.3). A diagram of their relationship is shown in Fig. 2. Portrait Stylization transforms a real user image into a stylized avatar image, keeping as much personal identity (glasses, hairs, colors, etc.) as possible, while simultaneously normalizing the face to look closer to an avatar rendering. Next, the Self-supervised Avatar Parameterization module regresses a relaxed avatar vector from the stylization latent code via a MLP based Mapper. Finally, the Avatar Vector Conversion module discretizes part of the relaxed avatar vector to meet the requirement of the graphics engine using an appearance-based search.

3.1 Portrait Stylization

Portrait Stylization transforms user images into stylized images close to our target domain. This stage of our pipeline occurs entirely within the 2D image domain. We adopt an encoder-decoder framework for the stylization task. A novel transfer learning approach is applied to a StyleGAN model [Karras et al. 2020], including W+ space transfer learning, using a normalized style exemplar set, and a loss function that supports these modifications.

W+ space transfer learning: We perform transfer learning directly from the W+ space, unlike previous methods [Gal et al. 2021; Song et al. 2021] where stylization transfer learning is done in the more entangled Z/Z+ space. The W+ space is more disentangled and can preserve more personal identity features. However, this design change introduces a challenge. We need to model a distribution prior $\mathcal{W}$ of the W+ space, as it is a highly irregular space [Wulff and Torralba 2020], and cannot be directly sampled like the Z/Z+ space (standard Gaussian distribution). We achieve this by inverting a large dataset of real face images into a W+ embeddings via a pre-trained image encoder [Tov et al. 2021], and then sample the latent codes from that prior. Fig. 3 provides one example of better preserved personalization. Notice that our method preserves glasses which are lost in the comparison method.

Normalized Style Exemplar Set: Our stylization method seeks to ignore pose and expression and produce a normalized image. In contrast, existing methods are optimized to preserve source to

(a) Input (b) AgileGAN (c) Our Stylization

Figure 3: Portrait stylization results. Compared with a state-of-the-art stylization method, AgileGAN [Song et al. 2021], our stylization does a better job at preserving the user's personal identity (e.g. glasses are preserved), and simultaneously normalizing the expressions (e.g. mouth is closed) for easier fitting in the downstream pipeline. ©Greg Mooney and Sebastiaan ter Burg.

(a) Stylization (b) Strict (c) Relaxed

Figure 4: Avatar Parameterization produces errors in final predictions if discrete types are enforced during training, such as hair and beard types in this example. Relaxing the discrete constraint allows easier optimization and thus better predictions which match the stylization target more closely.

target similarities literally, transferring specific facial expressions, head poses, and lighting conditions directly from user photos into target stylized images. This is not desirable for our later avatar parameterization stage as we are trying to extract the core personal identity features only. In order to produce normalized stylizations we limit the rendered exemplars provided during transfer learning to contain only neutral poses, expressions and illumination to ensure a good normalization. Fig. 3 provides an example of a smiling face. The comparison method preserves the smile, while our method successfully provides only the normalized core identity.

Loss: Our loss contains non-standard terms to support the needs of our pipeline. The target output stylization is not exactly aligned with the input due to pose normalization. Therefore, commonly used perceptual loss [Zhang et al. 2018] cannot be applied directly in decoder training. We instead use a novel segmented color loss.

The full objective comprises three loss terms to fine-tune the generator $\mathcal{G}_\phi$. Let $\mathcal{G}_{\phi_o}$ and $\mathcal{G}_{\phi_t}$ be the model before and after fine-tuning. We introduce a color matching loss at a semantic level. Specifically, we leverage two face segmentation models from BiSeNet [Yu et al. 2018] pre-trained on real and stylized data separately to match the color of semantic regions. Let $\mathbb{S} = \{hair, skin\}$ be the classes taken into consideration, and $\mathcal{B}^k(I)$ $(k \in \mathbb{S})$ be the mean color of pixels belonging to class k in image I. $\mathcal{B}^k_{real}$ and $\mathcal{B}^k_{style}$ represent real and stylized models separately. The semantic color matching loss is:

$$\mathcal{L}_{sem} = \mathbb{E}_{w\sim\mathcal{W}}[\sum_{k\in\mathbb{S}}\left(\left\|\mathcal{B}^k_{real}(\mathcal{G}_{\phi_o}(w)) - \mathcal{B}^k_{style}(\mathcal{G}_{\phi_t}(w))\right\|^2\right)] \tag{1}$$

An adversarial loss is used to match the distribution of the translated images to the target stylized set distribution $\mathcal{Y}$, where D is the StyleGAN2 discriminator [Karras et al. 2020]:

$$\mathcal{L}_{adv} = \mathbb{E}_{y\sim\mathcal{Y}}[\min(0, -1+D(y))] + \mathbb{E}_{w\sim\mathcal{W}}[\min(0, -1-D(\mathcal{G}_{\phi_t}(w)))] \tag{2}$$

Also, to improve training stability and prevent artifacts, we use R1 regularization [Mescheder et al. 2018] for the discriminator: $\mathcal{L}_{R1} = \frac{\gamma}{2}\mathbb{E}_{y\sim\mathcal{Y}}[\|\nabla D(y)\|^2]$, where we set $\gamma = 10$ empirically.

Finally, the generator and discriminators are jointly trained to optimize the combined objective $\min_\phi \max_D \mathcal{L}_{stylize}$, where

$$\mathcal{L}_{stylize} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{sem}\mathcal{L}_{sem} + \lambda_{R1}\mathcal{L}_{R1} \tag{3}$$

$\lambda_{adv} = 1, \lambda_{sem} = 12, \lambda_{R1} = 5$ are constant weights set empirically. Please see the appendix A for more details.

3.2 Self-supervised Avatar Parameterization

Avatar Parameterization finds a set of parameters for the rendering engine which produces an avatar matching the stylized portrait as closely as possible. We call the module which finds parameters the *mapper*. To facilitate training the mapper, we use a differentiable neural rendering engine we call the *imitator*.

A particular avatar is defined by an avatar vector with both continuous and discrete parameters. Continuous parameters are used to control primarily placement and size, for example eye size, eye rotation, mouth position, and head width. Discrete parameters are used to set individual assets and textures such as hair types, beard types, and skin tone textures. All parameters are concatenated into a vector with discrete parameters represented as one-hot vectors.

Mapper Training: The Mapper takes the results of portrait stylization as input and outputs an avatar vector which defines a similar looking avatar. Rather than using the stylized image itself as input, we use the latent code $w+$ derived from the stylization encoder, since it is a more compact representation and contains facial semantic styles from coarse to fine [Karras et al. 2019].

The Mapper is built as an MLP, and trained using a Mapper Loss which measures the similarity between the stylized image,

(a) Relaxed Result (b) Directly Classify (c) Our Conversion

Figure 5: Avatar Vector Conversion is necessary to convert the relaxed result produced during parameterization into discrete types suitable for the graphics engine. Direct classification often fails to select the best type. Our conversion selects the best match to the relaxed type by searching through all available discrete types. Notice in this example that the skin tone and hair type are much closer using our method.

$\mathcal{I}_{style}$, and the imitator output, $\mathcal{I}_{imitate}$. This loss function contains several terms to measure the global and local similarity.

To preserve global appearance, we incorporate identity loss $\mathcal{L}_{id}$ measuring the cosine similarity between two faces built upon a pretrained ArcFace [Deng et al. 2019a] face recognition network $\mathcal{R}$: $\mathcal{L}_{id} = 1 - cos(\mathcal{R}(I_{style}), \mathcal{R}(I_{imitate}))$. For a more fine-grained similarity measurement, LPIPS loss [Zhang et al. 2018] is adopted: $\mathcal{L}_{lpips} = \|\mathcal{F}(I_{style}) - \mathcal{F}(I_{imitate})\|_2$, where $\mathcal{F}$ denotes the perceptual feature extractor. Additionally, we use a color matching loss to obtain more faithful colors for the skin and hair region:

$$\mathcal{L}_{color} = \sum_{k \in \mathbb{S}} \left(\left\| \mathcal{B}^k_{style}(I_{style}) - \mathcal{B}^k_{style}(I_{imitate}) \right\|^2 \right) \quad (4)$$

The final loss function is:

$$\mathcal{L}_{mapper} = \lambda_{id}\mathcal{L}_{id} + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{color}\mathcal{L}_{color} \quad (5)$$

where $\lambda_{id} = 0.4$, $\lambda_{lpips} = 0.8$, $\lambda_{color} = 0.8$ are set empirically.

We empirically choose the best loss terms to provide good results. An ablation study of these terms is provided in the results section.

Differentiable Imitator: The imitator is a neural renderer trained to replicate the output of the graphics engine as closely as possible given an input avatar vector. The imitator has the important property of differentiablity, making it suitable for inclusion in an optimization framework. We leverage an existing neural model [Karras et al. 2019] as the backbone generator, which is capable of generating high quality avatar renderings. We train it with synthetic avatar data supervisedly. See the appendix B.1 for details.

Discrete Parameters: Solving for discrete parameters is challenging because of unstable convergence. Some methods handle this via quantization during optimization [Bengio et al. 2013; Cheng et al. 2018; Jang et al. 2016; Van Den Oord et al. 2017]. However, we found that quantization after optimization, which relaxes the discrete constraint during training and re-apply it as postprocessing, is more effective for our task. Below we describe the relaxed optimization and in Sec. 3.3 we present the quantization method.

Our solution to training discrete parameters in the mapper makes use of the imitator's interpolation property. When mixing two avatar vectors, the imitator still produces a valid rendering. That is, given the one-hot encoding v_1 and v_2 of two hair or beard types, their linear interpolation $\mathrm{v}_{mix} = (1-\alpha) \cdot \mathrm{v}_1 + \alpha \cdot \mathrm{v}_2$ ($\alpha \in [0, 1]$) produces a valid result. Please see the appendix B.1 for details.

Thus, when training the mapper we do *not* strictly enforce discrete parameters, and instead apply a softmax function to the final activation of the mapper to allow a continuous optimization space while still discouraging mixtures of too many asset types.

We compare our relaxed training with a strict training method performing quantization during optimization. In the forward pass, it quantizes the softmax result by picking the entry with maximum probability. In the backward pass, it back-propagates unaltered gradients in a straight-through way [Bengio et al. 2013]. In Fig. 4, our method produces a much closer match to the stylization results.

3.3 Avatar Vector Conversion

The graphics engine requires discrete inputs for attributes such as hair and glasses. However the mapper module in Avatar Parameterization produces continuous values. One straightforward approach for discretization is to pick the type with the highest probability given the softmax result. However, we observe that this approach does not achieve good results, especially when dealing with multi-class attributes (e.g. 45 hair types). The challenge is that the solution space is under-constrained. Medium length hair can be achieved by selecting the medium length hair type, or by mixing between short and long hair types. In the latter case, simply selecting the highest probability of short or long hair is clearly not optimal.

We discretize the relaxed avatar vector via searching over all candidates from the asset list for each attribute, while fixing all other parameters. Using the image result from the imitator $\mathcal{I}_{imitate}$ as target, we use the loss function from Eq. 5 as an objective to measure the similarity between $\mathcal{I}_{imitate}$ and the candidate result $\mathcal{I}_{cand}$. By minimizing the objective, we can find the best solution for each attribute. The selections for each attribute are combined to create the avatar vector used for graphics rendering and animation. Fig. 5 provides a comparison of direct classification and our method. Note that direct classification makes incorrect choices for hair type and skin color while ours closely matches the reference image.

4 EXPERIMENTAL ANALYSIS

Cascaded Domain Bridging: To illustrate the effect of each stage in the proposed three-stage pipeline, the intermediate results are visualized in Fig. 6. Notice how the three stages progressively bridge the domain gap between real images and stylized avatars. To measure how close the intermediate results are in comparison to the target avatar domain, we use the perceptual metric FID [Kilgour et al. 2019]. Notice that the FID becomes lower after each stage, demonstrating the gradual reduction of domain gap.

Visual Comparison with Baseline Methods: We compare the proposed method against a number of baselines, shown in Fig. 7. *CNN* is a naive supervised method using rendered avatar images to train a CNN [Sandler et al. 2018] to fit ground truth parameters. The CNN is then applied on the segmented head region of the input image. The domain gap causes the CNN to make poor predictions.

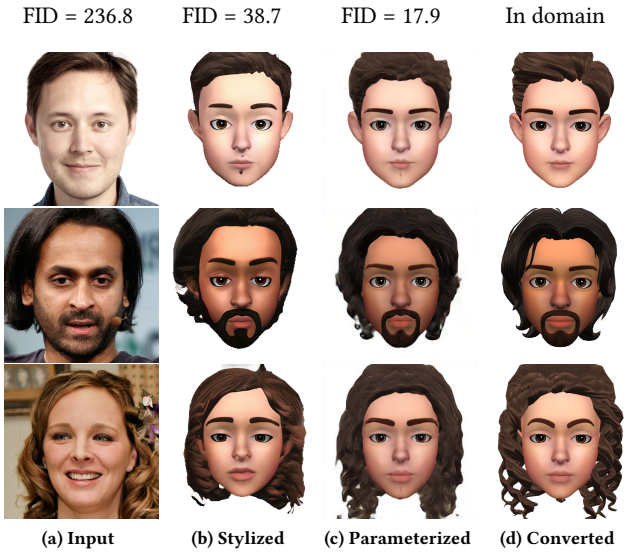


Figure 6: Progressive domain crossing. (b) At the portrait stylization stage, the images may still contain characteristics outside the domain of a graphics avatar, such as hair shape and non-frontal pose. (c) At the parameterization stage, the images are within the target domain, but may contain mixtures of components. (d) Finally, after vector conversion the output is a strict avatar vector which can be rendered by the graphics engine. Using FID as a measure of image distribution similarity, notice that each step brings us closer to the final target avatar domain. ©Marcin Wichary, TechCrunch and Vanity Productions.

Our stylization + CNN narrows the domain gap by applying the CNN to our stylized results. This noticeably improves predictions, however errors in hair and face coloration remain. Since the CNN is only trained on synthetic data, it cannot regress the parameters properly due to the domain gap between training and test data even for stylized images. *F2P [2019]* is a self-supervised optimization-based method designed for semi-realistic avatars. This method fails to do well, likely because it naively aligns the segmentation of real faces and the avatar faces, without considering the domain gap. *Manual* results were created by expert-trained users. Given a real face, the users were asked to build an avatar that preserves personal identity while demonstrating high attractiveness based on their own judgement. Visually, our method shows a quality similar to manual creation, demonstrating the utility of our method.

Numerical Comparison with Baseline Methods: To evaluate results numerically we rely on judgements made by human observers recruited through Amazon Mechanical Turk [3]. We conduct two user studies for quantitative evaluation: *Attribute Evaluation* and *Matching*. We perform attribute evaluation to evaluate whether users believe that specific identity attributes such as hair color and style match the source photograph using a yes/no selection. 330 opinions were collected for each of 6 attributes. Table 1 shows results, indicating that our method retains photograph identity better than the baseline. In the matching task, we evaluate whether an avatar retains personal identity overall. Four random and diverse images were used to create avatars, and the subject must choose which is the correct match to a specific photograph. A total of 990 judgements were collected. Avatars created with our method were identified correctly significantly more often than baseline methods, approaching the level of manually created avatars.

[3]https://www.mturk.com/

Table 1: Numerical results from two user studies. Our method is judged to produce better avatars than the baseline methods, approaching the quality of manual work. Attribute evaluation: judge whether a specific attribute of the created avatar matches the human image. Matching: choose the correct one out of four avatars which matches the human image.

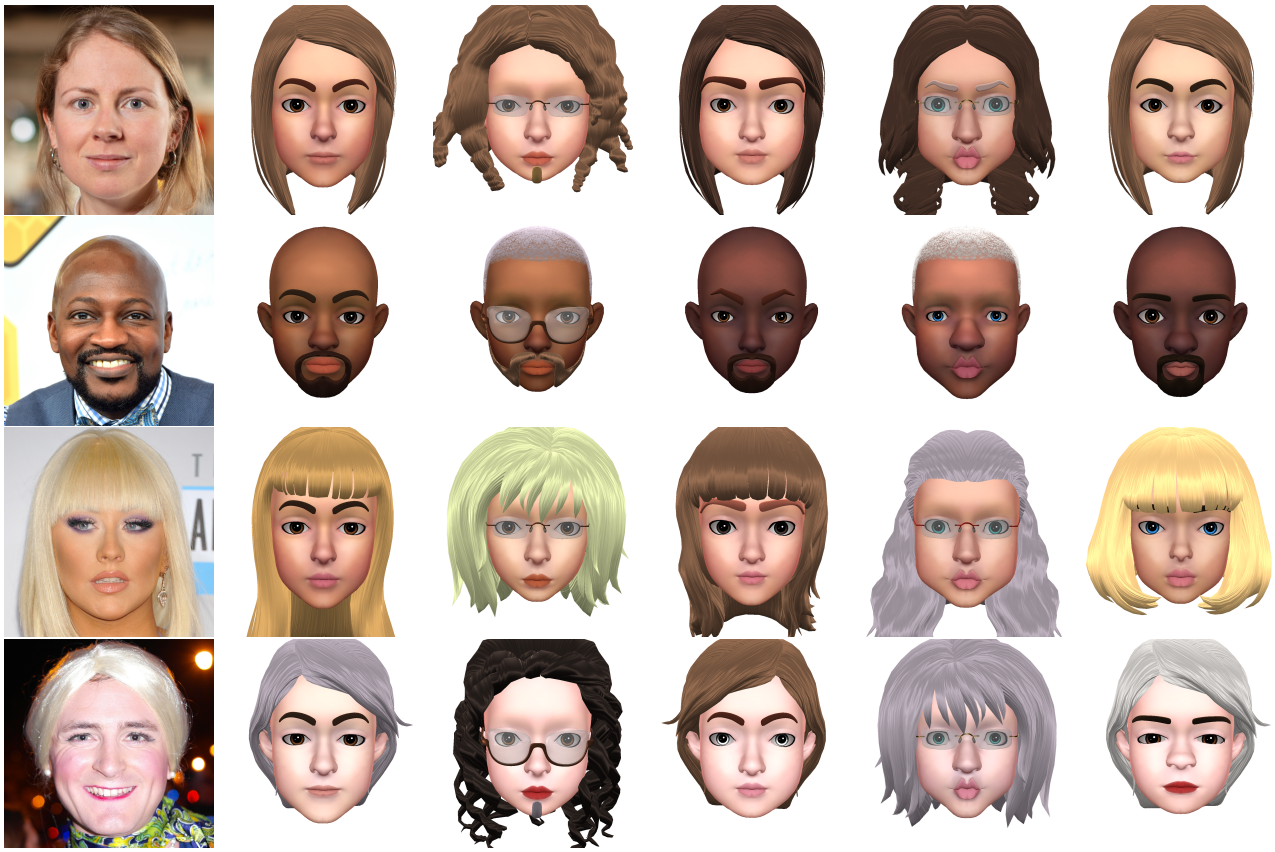
	Attribute Evaluation						Match Task
	beard type	face shape	brow type	hair color	hair style	skin tone	
F2P [2019]	0.36	0.46	0.22	0.21	0.12	0.36	0.67
CNN	0.17	0.54	0.22	0.46	0.30	0.50	0.57
Stylization+CNN	0.45	0.69	0.38	0.57	0.43	0.66	0.82
Ours	**0.82**	**0.94**	**0.88**	**0.82**	**0.72**	**0.82**	**0.92**
Manual	0.94	0.97	0.85	0.90	0.86	0.94	0.96

Table 2: Ablation study for mapper training losses. Users picked the best matching avatar from the six candidates produced by loss combinations. The scores show the fraction of each combination picked. $\mathcal{L}_{LPIPS}$ is the most significant component, while $\mathcal{L}_{id}$ and $\mathcal{L}_{color}$ also improve the results.

ID	LPIPS	ID+LPISP	ID+Color	LPIPS+Color	ID+LPIPS+Color
9.3%	17.8%	18.7%	14.8%	19.1%	**20.3%**

Portrait Stylization Ablation: To study the impact of Portrait Stylization on the complete avatar creation pipeline we compare three options, shown in Fig. 8. *No stylization* removes this stage entirely and uses the real image as input to parameterization loss calculation. Without stylization, the parameterization module tries to match the real image with the target stylized avatar, leading to poor visual quality. *AgileGAN* [Song et al. 2021] is a state-of-the-art stylization method. It provides stylization and thus improves the final avatar attractiveness compared to no stylization. However, it cannot remove the impact of expressions and does not handle glasses well. In Row 1 (b), the smile expression is explained as a big mouth in the fitting stage, and personal information like glasses is not preserved in Row 2 (b). Our method addresses these issues and achieves better results in both visual quality and personal identity.

Mapper Losses Ablation: To study the importance of including all losses while training the Mapper, we generate results using different permutations of loss terms (identity, LPIPS, color). We then collected 990 user judgements from Amazon Mechanical Turk, to select the best matching results to the input image among six permutation results. Table 2 shows the fraction of each option selected. The full set of losses achieves the best score by a small margin, matching



(a) Input (b) Ours (c) CNN (d) Stylization + CNN (e) F2P [2019] (f) Manual

Figure 7: Results comparison. (a) Given an input image, (b) our method produces an avatar in the target cartoon style that looks similar to the user. (c) A CNN trained on synthetic data produces incorrect beard, hair style, and glasses on real image inputs due to the significant domain gap. (d) Applying the CNN instead to the results of stylization reduces the domain gap and thus improves results, however significant errors remain. (e) F2P, a baseline method intended to produce semi-realistic avatars does not consider the domain gap and thus produces poor results when used with stylized avatars [Shi et al. 2019]. (f) Manual results were created by expert-trained users. Our results approximate the quality obtainable through manual creation. ©Sebastiaan ter Burg, NIGP, YayA Lee and S Pakhrin.

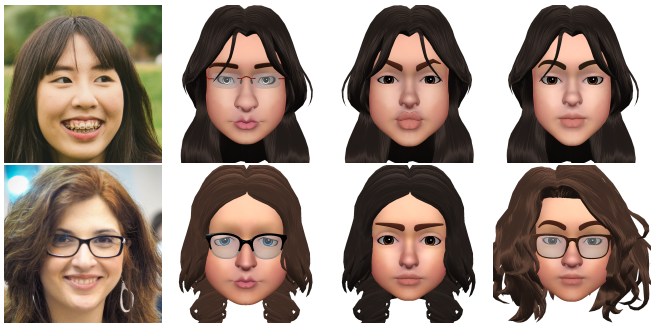
our observations that the overall method is robust to the precise selection of loss, but that the additional terms help in some cases.

5 LIMITATIONS

We observe two main limitations to our method. First, our method occasionally produces wrong predictions on assets covering a small area, because their contribution to the loss is small and gets ignored. The eye color in Fig. 9 (a) is an example of this difficulty. Redesigning the loss function might resolve this problem. Second, lighting is not fully normalized in the stylization stage, leading to incorrect skin tone estimates when there are strong shadows, shown in Fig. 9 (b). This problem could potentially be addressed by incorporating intrinsic decomposition into the pipeline. In addition to the limitations of our method, we experience a loss of ethnicity in the final results, which is mainly introduced by the graphics engine, as also evidenced by the manually-created results. This issue could be addressed by improving the diversity of the avatar system.

6 CONCLUSION

In summary, we present a self-supervised stylized avatar auto-creation method with cascaded domain crossing. Our method demonstrates that the gap between the real images domain and the target avatar domain can be progressively bridged with a three-stage pipeline: portrait stylization, self-supervised avatar parameterization, and avatar vector conversion. Each stage is carefully designed and cannot be simply removed. Experimental results show that our approach produces high quality attractive 3D avatars with personal identities preserved. In the future, we will extend the proposed pipeline to other domains, such as cubism and caricature avatars.



Input (a) No stylization (b) AgileGAN (c) Ours

Figure 8: We ablate by removing the stylization stage, as well as replacing our stylization with a state-of-the-art method. In each case the final renderings from the graphics engine are shown. (a) Fitting directly on a user image results in an avatar that lacks attractiveness. (b) Replacing our stylization with AgileGAN [2021] suffers from missing personal information such as glasses and artifacts where smiles are misinterpreted as heavy lips or mustache. (c) Our stylization retains personal features like glasses, and generate visually appealing results in spite of expressions. ©Chang-Ching Su and Luca Boldrini.

(a) Limitation - Small areas (b) Limitation - Shadows

Figure 9: Limitations: (a) failure on a parameter (eye color) affecting a small number of pixels. (b) incorrect skin tone prediction caused by shadows. ©Daniel Åberg and Peter Bright.

REFERENCES

Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. 2010. High-quality single-shot capture of facial geometry. In *ACM SIGGRAPH 2010 papers*. 1–9.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 187–194.

Hongrui Cai, Yudong Guo, Zhuang Peng, and Juyong Zhang. 2021. Landmark detection and 3D face reconstruction for caricature using a nonlinear parametric model. *Graphical Models* 115 (2021), 101103.

Chen Cao, Hongzhi Wu, Yanlin Weng, Tianjia Shao, and Kun Zhou. 2016. Real-time facial animation with image-based dynamic avatars. *ACM Transactions on Graphics* 35, 4 (2016).

Kaidi Cao, Jing Liao, and Lu Yuan. 2018. CariGANs: Unpaired Photo-to-Caricature Translation.

Zhixiang Chen and Tae-Kyun Kim. 2021. Learning Feature Aggregation for Deep 3D Morphable Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13164–13173.

Pengyu Cheng, Chang Liu, Chunyuan Li, Dinghan Shen, Ricardo Henao, and Lawrence Carin. 2018. Straight-through estimator as projected Wasserstein gradient flow. In *Neural Information Processing Systems (NeurIPS) Workshop*.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019a. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 2019b. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.

Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 2020. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–38.

Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. 2021. StyleGAN-NADA: CLIP-Guided Domain Adaptation of Image Generators. arXiv:2108.00946 [cs.CV]

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: a deep learning based sketching system for 3D face and caricature modeling. *ACM Transactions on graphics (TOG)* 36, 4 (2017), 1–12.

Xiaoguang Han, Kangcheng Hou, Dong Du, Yuda Qiu, Shuguang Cui, Kun Zhou, and Yizhou Yu. 2018. Caricatureshop: Personalized and photorealistic caricature sketching. *IEEE transactions on visualization and computer graphics* 26, 7 (2018), 2349–2361.

Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. 2017. Avatar digitization from a single image for real-time rendering. *ACM Transactions on Graphics (ToG)* 36, 6 (2017), 1–14.

Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. 2015. Dynamic 3D avatar creation from hand-held video input. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–14.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. 2019. Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms.. In *INTERSPEECH*. 2350–2354.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. 2020. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thomas Lewiner, Thales Vieira, Dimas Martínez, Adelailson Peixoto, Vinícius Mello, and Luiz Velho. 2011. Interactive 3D caricature from harmonic exaggeration. *Computers & Graphics* 35, 3 (2011), 586–595.

Song Li, Songzhi Su, Juncong Lin, Guorong Cai, and Li Sun. 2021. Deep 3D caricature face generation with identity and structure consistency. *Neurocomputing* 454 (2021), 178–188.

Jiangke Lin, Yi Yuan, and Zhengxia Zou. 2021. MeInGame: Create a Game Character Face from a Single Portrait. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 311–319.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*. 3730–3738.

Huiwen Luo, Koki Nagano, Han-Wei Kung, Qingguo Xu, Zejian Wang, Lingyu Wei, Liwen Hu, and Hao Li. 2021. Normalized Avatar Synthesis Using StyleGAN and Perceptual Refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11662–11672.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which training methods for GANs do actually converge?. In *International conference on machine learning*. PMLR, 3481–3490.

Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. 2020. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*. Springer, 319–345.

Weilong Peng, Zhiyong Feng, Chao Xu, and Yong Su. 2017. Parametric t-spline face morphable model for detailed fitting in shape subspace. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6139–6147.

Justin NM Pinkney and Doron Adler. 2020. Resolution dependent gan interpolation for controllable image synthesis between domains. *arXiv preprint arXiv:2010.05334* (2020).

Yuda Qiu, Xiaojie Xu, Lingteng Qiu, Yan Pan, Yushuang Wu, Weikai Chen, and Xiaoguang Han. 2021. 3dcaricshop: A dataset and a baseline method for single-view 3d caricature face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10236–10245.

Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. 2021. Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

Tianyang Shi, Yi Yuan, Changjie Fan, Zhengxia Zou, Zhenwei Shi, and Yong Liu. 2019. Face-to-parameter translation for game character auto-creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 161–170.

Tianyang Shi, Zhengxia Zuo, Yi Yuan, and Changjie Fan. 2020. Fast and Robust Face-to-Parameter Translation for Game Character Auto-Creation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1733–1740.

Guoxian Song, Linjie Luo, Jing Liu, Wan-Chun Ma, Chunpong Lai, Chuanxia Zheng, and Tat-Jen Cham. 2021. AgileGAN: stylizing portraits by inversion-consistent transfer learning. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. 2021. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).

Roberto C Cavalcante Vieira, Creto A Vidal, and Joaquim Bento Cavalcante-Neto. 2013. Three-dimensional face caricaturing by anthropometric distortions. In *2013 XXVI Conference on Graphics, Patterns and Images*. IEEE, 163–170.

Qianyi Wu, Juyong Zhang, Yu-Kun Lai, Jianmin Zheng, and Jianfei Cai. 2018. Alive caricature from 2d to 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7336–7345.

Jonas Wulff and Antonio Torralba. 2020. Improving Inversion and Generation Diversity in StyleGAN using a Gaussianized Latent Space. In *Conference on Neural Information Processing Systems*.

Sicheng Xu, Jiaolong Yang, Dong Chen, Fang Wen, Yu Deng, Yunde Jia, and Xin Tong. 2020. Deep 3d portrait from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7710–7720.

Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. 2020. FaceScape: A Large-Scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zipeng Ye, Mengfei Xia, Yanan Sun, Ran Yi, Minjing Yu, Juyong Zhang, Yu-Kun Lai, and Yong-Jin Liu. 2021. 3D-CariGAN: an end-to-end solution to 3D caricature generation from normal face photos. *IEEE Transactions on Visualization and Computer Graphics* (2021).

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

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. 2021. Mind the Gap: Domain Gap Control for Single Shot Domain Adaptation for Generative Adversarial Networks. arXiv:2110.08398 [cs.CV]

Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. 2018. State of the art on monocular 3D face reconstruction, tracking, and applications. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 523–550.

A PORTRAIT STYLIZATION DETAILS

Segmentation Models: The avatar segmentation model is trained using 20k randomly sampled avatar vectors with neural pose, expression and illumination. For real image segmentation, we used an open-source pre-trained BiSeNet module[4] [Yu et al. 2018].

Distribution Prior $\mathcal{W}$: To sample W+ distribution prior, we inverse CelebA dataset [Liu et al. 2015] into W+ space using a pre-trained e4e encoder [Tov et al. 2021].

[4]https://github.com/zllrunning/face-parsing.PyTorch

Normalized Style Exemplar Set $\mathcal{Y}$: For training stylized generator $\mathcal{G}_{\phi_t}$, we synthetically rendered a diverse set of 150 avatar imageries with normalized facial expressions.

B AVATAR PARAMETERIZATION DETAILS

B.1 Imitator

To train our module in a self-supervised way, we plug-in a differentiable neural renderer (i.e. imitator) in our learning framework. As we mentioned in the main paper, the imitator can take a relaxed avatar vector as input, although the imitator itself is trained with strict avatar vector. No matter the input is a relaxed or strict avatar vector, it can produce a valid rendering. In this way, we can supervise the training in image space without any ground-truth for the parameters. Due to the differentiability of the imitator, the parameterization stage can be trained with gradient descent. To achieve high fidelity rendering quality, we leverage the StyleGAN2 generator [Karras et al. 2019] as our backbone, which is capable of generating high quality renderings matching the graphics engine. The imitator consists of an encoder $\mathcal{E}_i$ implemented using MLP and a generator $\mathcal{G}_i$ adopted from StyleGAN2. The encoder translates an input avatar vector to a latent code w+. The generator then produces a high-quality image given the latent code.

Training: In order to fully utilize the image generation capability of StyleGAN2, we propose to train the imitator in two steps: 1) we first train a StyleGAN2 from scratch with random rendering samples generated by our graphics engine to obtain a high-quality image generator, without any label or conditions; then 2) we train the encoder and the generator together with images and corresponding labels, result in a conditional generator. Given an avatar vector v, a target image $\mathcal{I}_{gt}$ and the generated image $\mathcal{I}_{gen} = \mathcal{G}_i(\mathcal{E}_i(v))$, we use the following loss function combination to perform the second step training:

$$\mathcal{L}_{imitator} = \lambda_1 \|\mathcal{I}_{gen} - \mathcal{I}_{gt}\|_1 + \lambda_2 \mathcal{L}_{lpips} + \lambda_3 \mathcal{L}_{id} \tag{6}$$

where the first term is an L1 loss, which encourages less blurring than L2. In addition, $\mathcal{L}_{lpips}$ is the LPIPS loss adopted from [Zhang et al. 2018],

$$\mathcal{L}_{lpips} = \|\mathcal{F}(I_1) - \mathcal{F}(I_2)\|_2 \tag{7}$$

where $\mathcal{F}$ denotes the perceptual feature extractor. $\mathcal{L}_{id}$ is the identity loss which measures the cosine similarity between two faces built upon a pretrained ArcFace [Deng et al. 2019a] face recognition network $\mathcal{R}$,

$$\mathcal{L}_{id} = 1 - cos(\mathcal{R}(I_1), \mathcal{R}(I_2)) \tag{8}$$

We set $\lambda_1 = 1.0$, $\lambda_2 = 0.8$, $\lambda_3 = 1.0$, empirically.

Interpolation property: Fig. 10 provides an example of the interpolation property of the imitator which enables relaxed optimization over the discrete parameters.

Implementation: To train the imitator, we randomly generate 100,000 images and corresponding parameters. Note that although random sampling leads to strange avatars, our imitator can generate images matching the graphics engine well by seeing plenty of

Figure 10: Interpolation of avatar vectors. The neural rendering imitator which temporarily replaces the traditional graphics engine is differentiable, allowing the relaxation of the strict constraint on discrete types. Linear interpolation between two avatar vectors results in the gradual disappearance of the beard and the gradual growth of the hair.

samples in the parameter space. Please refer to our supplementary video for a side-by-side comparison.

We train StyleGAN2 using the official source code[5] with images of size $256 \times 256 \times 3$, thus the latent code $w+$ has a shape of 14×512. We build the encoder $\mathcal{E}_i$ with 14 individual small MLPs, each is responsible for mapping from the input vector to one latent style. Given the pretrained generator, we train the encoder and simultaneously finetune the generator with Adam [Kingma and Ba 2015]. We set the initial learning as 0.01 and decay it by 0.5 each two epochs. In our experiments, it takes around 20 epochs to converge.

B.2 Mapper

We use CelebA-HQ [Lee et al. 2020] and FFHQ [Karras et al. 2019] as our training data. To collect a high quality dataset for training, we use the Azure Face API [6] to analyze the facial attributes and keep only facial images that meet our requirements:

1) within a limited pose range ($yaw < 8°, pitch < 8°, roll < 5°$)
2) without headwears
3) without extreme expressions
4) without any occlusions

Finally, we collect 21,522 images in total for mapper training.

The input is an 18×512 latent code taken from the Stylization module. Each one of the 18 layers latent code is passed to an individual MLP. The output features are then concatenated together. After that, we apply two MLP heads to generate continuous and discrete parameters separately.

We apply a scaling before the softmax function for discrete parameters:

$$\mathcal{S}(x) = \frac{e^{\beta x_k}}{\Sigma_{i=1}^{N} e^{\beta x_i}}, k = 1, ...N \tag{9}$$

where $\beta > 1$ is a coefficient that performs non-maximum suppression over some types that contribute less than the dominant ones, and N is the number of discrete types. During training, we gradually increase the coefficient β to perform an easy-to-hard training by decreasing the smoothness. Empirically, we increase β by 1 for each epoch. We train the mapper for 20 epochs.

[5] https://github.com/NVlabs/stylegan2-ada-pytorch
[6] https://azure.microsoft.com/en-us/services/cognitive-services/face