3DAvatarGAN: Bridging Domains for Personalized Editable Avatars

Rameen Abdal[†1]   Hsin-Ying Lee[2]   Peihao Zhu[†1]   Menglei Chai[2]   Aliaksandr Siarohin[2]

Peter Wonka[1]   Sergey Tulyakov[2]

[1]KAUST   [2]Snap Inc.
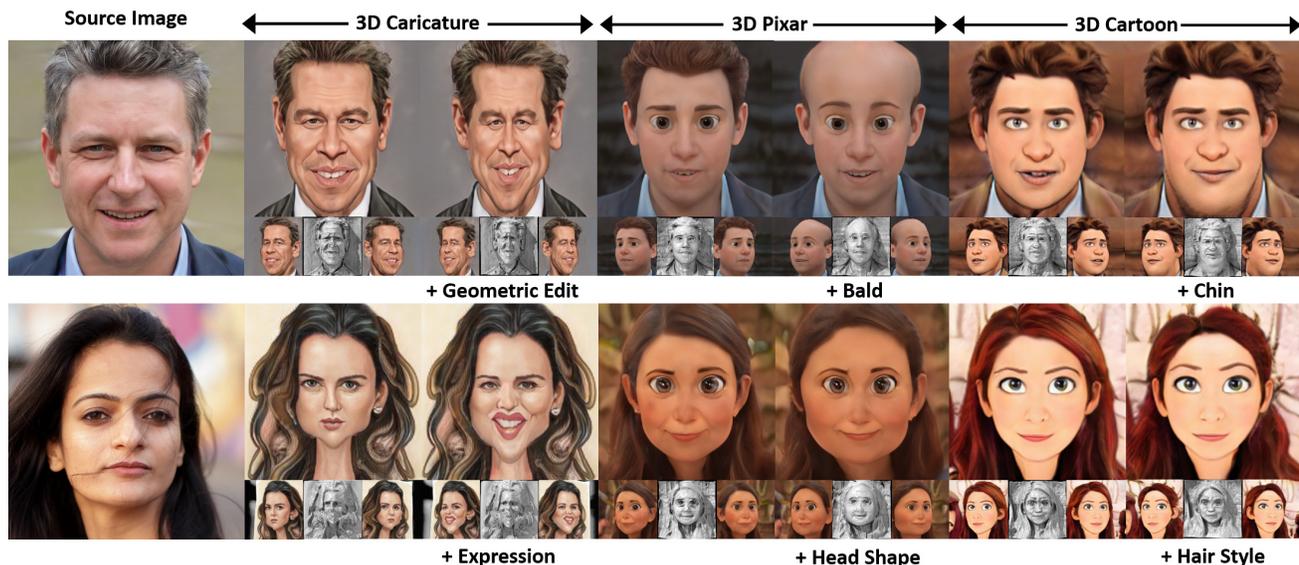
Figure 1. **Editable 3D avatars.** We present 3DAvatarGAN, a 3D GAN able to produce and edit personalized 3D avatars from a single photograph (real or generated). Our method distills information from a 2D-GAN trained on 2D artistic datasets like Caricatures, Pixar toons, Cartoons, Comics *etc.* and requires no camera annotations.

Abstract

Modern 3D-GANs synthesize geometry and texture by training on large-scale datasets with a consistent structure. Training such models on stylized, artistic data, with often unknown, highly variable geometry, and camera information has not yet been shown possible. Can we train a 3D GAN on such artistic data, while maintaining multi-view consistency and texture quality? To this end, we propose an adaptation framework, where the source domain is a pre-trained 3D-GAN, while the target domain is a 2D-GAN trained on artistic datasets. We, then, distill the knowledge from a 2D generator to the source 3D generator. To do that, we first propose an optimization-based method to align the distributions of camera parameters across domains. Second, we propose regularizations necessary to learn high-quality texture, while avoiding degenerate geometric solutions, such as flat shapes. Third, we show a deformation-based technique for modeling exaggerated geometry of artistic domains, enabling—as a byproduct—personalized geometric editing. Finally, we propose a novel inversion method for 3D-GANs linking the latent spaces of the source and the target domains. Our contributions—for the first time—allow for the generation, editing, and animation of personalized artistic 3D avatars on artistic datasets. Project Page: https://rameenabdal.github.io/3DAvatarGAN

1. Introduction

Photo-realistic portrait face generation is an iconic application demonstrating the capability of generative models especially GANs [28,30,31]. A recent development has witnessed an advancement from straightforwardly synthesizing 2D images to learning 3D structures without 3D supervision, referred to as 3D-GANs [10,41,55,64]. Such training

---

† Part of the work was done during an internship at Snap Inc.

is feasible with the datasets containing objects with highly consistent geometry, enabling a 3D-GAN to learn a distribution of shapes and textures. In contrast, artistically stylized datasets [25, 65] have arbitrary exaggerations of both geometry and texture, for example, the nose, cheeks, and eyes can be arbitrarily drawn, depending on the style of the artist as well as on the features of the subject, see Fig. 1. Training a 3D-GAN on such data becomes problematic due to the challenge of learning such an arbitrary distribution of geometry and texture. In our experiments (Sec. 5.1), 3D-GANs [10] generate flat geometry and become 2D-GANs essentially. A natural question arises, whether a 3D-GAN can synthesize consistent novel views of images belonging to artistically stylized domains, such as the ones in Fig. 1.

In this work, we propose a domain-adaption framework that allows us to answer the question positively. Specifically, we fine-tune a pre-trained 3D-GAN using a 2D-GAN trained on a target domain. Despite being well explored for 2D-GANs [25, 65], existing domain adaptation techniques are not directly applicable to 3D-GANs, due to the nature of 3D data and characteristics of 3D generators.

The geometry and texture of stylized 2D datasets can be arbitrarily exaggerated depending on the context, artist, and production requirements. Due to this, no reliable way to estimate camera parameters for each image exists, whether using an off-the-shelf pose detector [72] or a manual labeling effort. To enable the training of 3D-GANs on such challenging datasets, we propose three contributions. ① An optimization-based method to align distributions of camera parameters between domains. ② Texture, depth, and geometry regularizations to avoid degenerate, flat solutions and ensure high visual quality. Furthermore, we redesign the discriminator training to make it compatible with our task. We then propose ③ a *Thin Plate Spline (TPS)* 3D deformation module operating on a tri-plane representation to allow for certain large and sometimes extreme geometric deformations, which are so typical in artistic domains.

The proposed adaptation framework enables the training of 3D-GANs on complex and challenging artistic data. The previous success of domain adaptation in 2D-GANs unleashed a number of exciting applications in the content creation area [25, 65]. Given a single image such methods first find a latent code corresponding to it using GAN inversion, followed by latent editing producing the desired effect in the image space. Compared to 2D-GANs, the latent space of 3D-GANs is more entangled, making it more challenging to link the latent spaces between domains, rendering the existing inversion and editing techniques not directly applicable. Hence, we take a step further and explore the use of our approach to 3D artistic avatar generation and editing. Our final contribution to enable such applications is ④ a new inversion method for coupled 3D-GANs.

In summary, the proposed domain-adaption framework

allows us to train 3D-GANs on challenging artistic datasets with exaggerated geometry and texture. We call our method 3DAvatarGAN as it—for the first time—offers generation, editing, and animation of personalized stylized, artistic avatars obtained from a single image. Our results (See Sec. 5.2) show the high-quality 3D avatars possible by our method compared to the naive fine-tuning.

2. Related Work

GANs and Semantic Image Editing. Generative adversarial Networks (GANs) [19, 47] are one popular type of generative model, especially for smaller high-quality datasets such as FFHQ [32], AFHQ [14], and LSUN objects [67]. For these datasets, StyleGAN [28, 30, 32] can be considered as the current state-of-the-art GAN [27, 28, 30, 32, 33]. The disentangled latent space learned by StyleGAN has been shown to exhibit semantic properties conducive to semantic image editing [1, 3, 16, 22, 36, 44, 51, 56, 62]. CLIP [46] based image editing [2, 17, 44] and domain transfer [15, 70] are another set of works enabled by StyleGAN.

GAN Inversion. Algorithms to project existing images into a GAN latent space are a prerequisite for GAN-based image editing. There are mainly two types of methods to enable such a projection: optimization-based methods [1,13,57,71] and encoder-based methods [5,7,48,58,69]. On top of both streams of methods, the generator weights can be further modified after obtaining initial inversion results [49].

Learning 3D-GANs with 2D Data. Previously, some approaches attempt to extract 3D structure from pre-trained 2D-GANs [42, 52]. Recently, inspired by Neural Radiance Field (NeRF) [9, 37, 43, 68], novel GAN architectures have been proposed to combine implicit or explicit 3D representations with neural rendering techniques [11, 12, 20, 39–41, 50, 53, 55, 63, 64]. In our work, we build on EG3D [11] which has current state-of-the-art results for human faces trained on the FFHQ dataset.

Avatars and GANs. To generate new results in an artistic domain (*e.g.* anime or cartoons), a promising technique is to fine-tune an existing GAN pre-trained on photographs, e.g. [45, 54, 60]. Data augmentation and freezing lower layers of the discriminator are useful tools when fine-tuning a 2D-GAN [28, 38]. One branch of methods [18, 44, 70] investigates domain adaptation if only a few examples or only text descriptions are available. While others focus on matching the distribution of artistic datasets with diverse shapes and styles. Our work also falls in this domain. Among previous efforts, StyleCariGAN [25] proposes invertible modules in the generator to train and generate caricatures from real images. DualStyleGAN [65] learns two mapping networks in StyleGAN to control the style and structure of the new domain. Some works are trained on 3D data or require heavy labeling/engineering [21, 26, 66] and use 3D morphable models to map 2D images of carica-

Naïve Fine-Tuning   **Our Method**
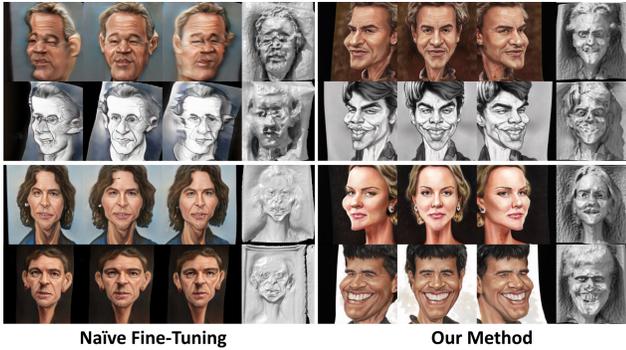
Figure 2. **Comparison with naive fine-tuning.** Comparison of generated 3D avatars with a naïvely fine-tuned generator G_{base} (left sub-figures) versus our generator G_{t} (right sub-figures). The corresponding sub-figures show comparisons in terms of texture quality (top two rows) and geometry (bottom two rows). See Sec. 5.1 for details.

tures to 3D models. However, such models fail to model the hair, teeth, neck, and clothes and suffer in texture quality. In this work, we are the first to tackle the problem of domain adaption of 3D-GANs and to produce fully controllable 3D Avatars. We employ 2D to 3D domain adaptation and distillation and make use of synthetic 2D data from StyleCari-GAN [25] and DualStyleGAN [65].

3. Domain Adaptation for 3D-GANs

The goal of domain adaptation for 3D-GANs is to adapt (both texture and geometry) to a particular style defined by a 2D dataset (Caricature, Anime, Pixar toons, Comic, and Cartoons [24, 25, 65] in our case). In contrast to 2D-StyleGAN-based fine-tuning methods that are conceptually simpler [29, 45], fine-tuning a 3D-GAN on 2D data introduces challenges in addition to domain differences, especially on maintaining the texture quality while preserving the geometry. Moreover, for these datasets, there is no explicit shape and camera information. We define the domain adaptation task as follows: Given a prior 3D-GAN *i.e.* EG3D (G_{s}) of source domain (T_{s}), we aim to produce a 3D Avatar GAN (G_{t}) of the target domain (T_{t}) while maintaining the semantic, style, and geometric properties of G_{s}, and at the same time preserving the identity of the subject between the domains ($T_{\text{s}} \leftrightarrow T_{\text{t}}$). Refer to Fig. 4 in supplementary for the pipeline figure. We represent G_{2D} as a teacher 2D-GAN used for knowledge distillation fine-tuned on the above datasets. Note that as T_{t} is not assumed to contain camera parameter annotations, the training scheme must suppress artifacts such as low-quality texture under different views and flat geometry (See Fig. 2). In the following, we discuss the details of our method.

3.1. How to align the cameras?

Selecting appropriate ranges for camera parameters is of paramount importance for high-fidelity geometry and texture detail. Typically, such parameters are empirically estimated, directly computed from the dataset using an off-the-shelf pose detector [10], or learned during training [8]. In domains we aim to bridge, such as caricatures for which a 3D model may not even exist, directly estimating the camera distribution is problematic and, hence, is not assumed by our method. Instead, we find it essential to ensure that the camera parameter distribution is consistent across the source and target domains. For the target domain, we use StyleGAN2 trained on FFHQ, fine-tuned on artistic datasets [25, 65]. Assuming that the intrinsic parameters of all the cameras are the same, we aim to match the distribution of extrinsic camera parameters of G_{s} and G_{2D} and train our final G_{t} using it (see illustration in Fig. 2 of the supplementary materials). To this end, we define an optimization-based method to match the sought distributions. The first step is to identify a canonical pose image in G_{2D}, where the yaw, pitch, and roll parameters are zero. According to Karras et al., [31], the image corresponding to the mean latent code satisfies this property. Let θ, ϕ be the camera Euler angles in a spherical coordinate system, r, c be the radius of the sphere and camera lookat point, and, M be a function that converts these parameters into the camera-to-world matrix. Let $I_{\text{s}}(w, \theta, \phi, c, r) = G_{\text{s}}(w, \text{M}(\theta, \phi, c, r))$ and $I_{\text{2D}}(w) = G_{\text{2D}}(w)$ represent an arbitrary image generated by G_{s} and G_{2D}, respectively, given the w code variable. Let k_{d} be the face key-points detected by the detector K_{d} [72], then

$$(c', r') := \underset{(c,r)}{\arg\min} \, \text{L}_{\text{kd}}(I_{\text{s}}(w'_{\text{avg}}, 0, 0, c, r), I_{\text{2D}}(w_{\text{avg}})),$$
(1)

where $\text{L}_{\text{kd}}(I_1, I_2) = \|k_{\text{d}}(I_1) - k_{\text{d}}(I_2)\|_1$ and w_{avg} and w'_{avg} are the mean w latent codes of G_{2D} and G_{s}, respectively. In our results, r' is determined to be 2.7 and c' is approximately $[0.0, 0.05, 0.17]$. The next step is to determine a safe range of the θ and ϕ parameters. Following prior works, StyleFlow [3] and FreeStyleGAN [35] (see Fig.5 of the paper), we set these parameters as $\theta' \in [-0.45, 0.45]$ and $\phi' \in [-0.35, 0.35]$ in radians.

3.2. What loss functions and regularizers to use?

Next, although the camera systems are aligned, the given dataset may not stem from a consistent 3D model, *e.g.*, in the case of caricatures or cartoons. This entices the generator G_{t} to converge to an easier degenerate solution with a flat geometry. Hence, to benefit from the geometric prior of G_{s}, another important step is to design the loss functions and regularizers for a selected set of parameters to update in G_{t}. Next, we discuss these design choices:

Figure 3. **Domain adaptation.** Domain adaptation results of images from source domain T_s (top row in each sub-figure) to target domain T_t. Rows two to five show corresponding 3D avatar results from different viewpoints.

Loss Functions. To ensure texture quality and diversity, we resort to the adversarial loss used to fine-tune GANs as our main loss function. We use the standard non-saturating loss to train the generator and discriminator networks used in EG3D [11]. We also perform lazy density regularization to ensure consistency of the density values in the final fine-tuned model G_t.

Texture Regularization. Since the texture can be entangled with the geometry information, determining which layers to update is important. To make use of the fine-style information encoded in later layers, it is essential to update the *tRGB* layer parameters (outputting tri-plane features) before the neural rendering stage. *tRGB* are convolutional layers that transform feature maps to 3 channels at each resolution (96 channels in triplanes). Moreover, since the network has to adapt to a color distribution of T_t, it is essential to update the decoder (*MLP* layers) of the neural rendering pipeline as well. Given the EG3D architecture, we also update the super-resolution layer parameters to ensure the coherency between the low-resolution and high-resolution outputs seen by the discriminator D.

Geometry Regularization. In order to allow the network to learn the structure distribution of T_t and at the same time ensure properties of $\mathcal{W}$ and $\mathcal{S}$ latent spaces are preserved, we update the earlier layers with regularization. This also encourages the latent spaces of T_s and T_t to be easily linked. Essentially, we update the deviation parameter Δs from the

s activations of the $\mathcal{S}$ space [62]. The s activations are predicted by $A(w)$, where A is the learned affine function in EG3D. The s activations scale the kernels of a particular layer. In order to preserve the identity as well as geometry such that the optimization of Δs does not deviate too far away from the original domain T_s, we introduce a regularizer given by

$$R(\Delta s) := \|\Delta s\|_1. \tag{2}$$

Note that we apply $R(\Delta s)$ regularization in a lazy manner, *i.e.*, with density regularization. Interestingly, after training, we can interpolate between s and $s + \Delta s$ parameters to interpolate between the geometries of samples in T_s and T_t (See Fig. 5).

Depth Regularization. Next, we observe that even though the above design choice produces better geometry for T_t, some samples from G_t can still lead to flatter geometry, and it is hard to detect these cases. We found that the problem is related to the relative depth of the background to the foreground. To circumvent this problem, we use an additional regularization where we encourage the average background depth of G_t to be similar to G_s. Let S_b be a face background segmentation network [34]. We first compute the average background depth of the samples given by G_s. This average depth is given by

$$a_d := \frac{1}{M} \sum_{n=1}^{M} (\frac{1}{N_n} \|D_n \odot S_b(I_n)\|_F^2). \tag{3}$$

4555
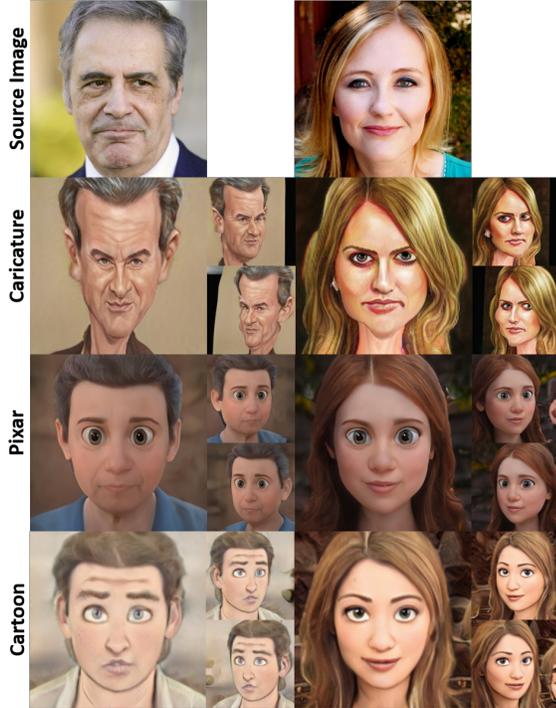
Figure 4. **3D avatars from real images.** Projection of real images on the 3D avatar generators.

Here, D_n is the depth map of the image I_n sampled from G_s, $\odot$ represents the *Hadamard* product, M is the number of the sampled images, and N_n is the number of background pixels in I_n. Finally, regularization is defined as:

$$\mathrm{R}(D) := \|a_d \cdot J - (D_t \odot \mathrm{S}_b(I_t))\|_F, \qquad (4)$$

where D_t is the depth map of the image I_t sampled from G_t and J is the matrix of ones having the same spatial dimensions as D_t.

3.3. What discriminator to use?

Given that the data in T_s and T_t is not paired and T_t is not assumed to contain camera parameter annotations, the choice of the discriminator (D) used for this task is also a critical design choice. Essentially, we use the unconditional version of the dual discriminator proposed in EG3D, and hence, we do not condition the discriminator on the camera information. As a result, during the training, G_t generates arbitrary images with pose using $\mathrm{M}(\theta', \phi', c', r')$, and the discriminator discriminates these images using arbitrary images from T_t. We train the discriminator from scratch and in order to adapt $T_s \rightarrow T_t$, we use the StyleGAN-ADA [28] training scheme and use R1 regularization.

3.4. How to incorporate larger geometric deformations between domains?

While the regularizers are used to limit the geometric changes when adapting from T_s to T_t, modeling large ge-

ometric deformations, e.g., in the caricature dataset is another challenge. One choice to edit the geometry is to use the properties of tri-plane features learned by EG3D. We start out by analyzing these three planes in G_s. We observe that the frontal plane encodes most of the information required to render the final image. To quantify this, we sample images and depth maps from G_s and swap the front and the other planes from two random images. Then we compare the difference in *RGB* values of the images and the *Chamfer* distance of the depth maps. While swapping the frontal tri-planes, the final images are completely swapped, and the *Chamfer* distance changes by $80 \sim 90\%$ matching the swapped image depth map. In the case of the other two planes, the *RGB* image is not much affected and the *Chamfer* distance of the depth maps is reduced by only $20 \sim 30\%$ in most cases.

Given the analysis, we focus to manipulate the *2D* front plane features to learn additional deformation or exaggerations. We learn a *TPS (Thin Plate Spline)* [61] network on top of the front plane. Our *TPS* network is conditioned both on the front plane features as well as the $\mathcal{W}$ space to enable multiple transformations. The architecture of the module is similar to the standard StyleGAN2 layer with an *MLP* appended at the end to predict the control points that transform the features. Hence, as a byproduct, we also enable 3D-geometry editing guided by the learned latent space. We train this module separately after G_t has been trained. We find that joint training is unstable due to exploding gradients arising from the large domain gap between T_s and T_t in the initial stages. Formally, we define this transformation as:

$$\mathrm{T}(w, f) := \Delta c, \qquad (5)$$

where, w is the latent code, f is the front plane, and c are the control points.

Let c_I be the initial control points producing an identity transformation, (c_1, c_2) be the control points corresponding to front planes (f_1, f_2) sampled using $\mathcal{W}$ codes (w_1, w_2), respectively, and (c_1', c_2') be points with (w_1, w_2) swapped in the *TPS* module. To regularize and encourage the module to learn different deformations, we have

$$\mathrm{R}(\mathrm{T}_1) := \alpha \sum_{n=1}^{2} \|c_I - c_n\|_1 - \beta\|c_1 - c_2\|_1 - \sigma\|c_1' - c_2'\|_1. \qquad (6)$$

We use initial control point regularization to regularize large deviations in the control points which would otherwise explode. Additionally, to learn extreme exaggerations in T_t and *'in expectation'*, conform to the target distribution in the dataset, we add an additional loss term. Let $\mathrm{S}(I)$ be the soft-argmax output of the face segmentation network [34] given an image I and assuming that S generalizes to caricatures, then

$$\mathrm{R}(\mathrm{T}_2) := \|\mathrm{S}(G_t(w)), \mathrm{S}(I_t)\|_1 \qquad (7)$$
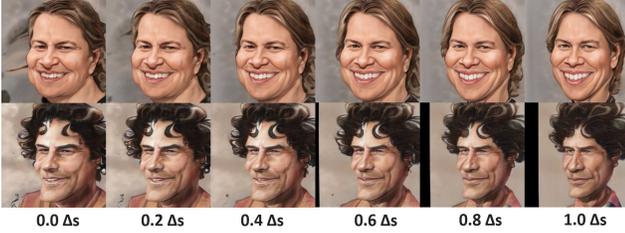
Figure 5. **Interpolation of** Δs. Geometric deformation using the interpolation of learned Δs parameters.

Eq. 6, Eq. 7, and adversarial training loss are used to train the *TPS* module. We adopt gradient clipping to make sure that the training does not diverge. See the illustrations in Fig. 3 and Fig. 4 of the supplementary materials.

4. Personalized Avatar Generation and Editing

Although 3D domain adaptation adapts $T_s \leftrightarrow T_t$, it is still a challenge to effectively link the latent spaces of G_s and G_t to generate personalized 3D avatars using a single photograph as the reference image. Particularly, the challenge arises due to the discrepancy in the coupled latent spaces when dealing with the projection of real photographs on 3D generators. Moreover, one would like to edit and animate these 3D avatars.

Projection. The task is to project a real image into the latent space of G_s, transfer the latent to G_t, and further optimize it to construct a 3D avatar. First, we use an optimization-based method to find the w code that minimizes the similarity between the generated and the real image in G_s. To achieve this, the first step is to align the cameras. We follow the steps mentioned in Sec. 3.1 for this step. Next, we use pixel-wise *MSE* loss and *LPIPS* loss to project the image into G_s [1]. Additionally, to preserve the identity of the subject, we use attribute classifiers *e.g.* caricature dataset [24] provides the coupled attribute information of real images and caricatures. We use such attribute classifier [24, 25] in a post-hoc manner as we notice that such networks can affect the texture in the target domain and could degenerate to narrow style outputs if applied during training. Moreover, such networks may not be available for all target domains. To avoid overfitting into G_s and encourage the easier transfer of the optimized latent code to G_t, we use $\mathcal{W}$ space optimization for this step. Finally, we initialize this w code for G_t and use additional attribute classifier loss [25] for T_t domain along with Depth regularization $R(D)$ (Eq. 4). As an approximation, we assume that attribute classifier [24, 25] generalizes across all domains. We use $\mathcal{W}/\mathcal{W}+$ space optimization to control the quality and diversity of the outputs. See Algorithm 1 in supplementary for the description.

Editing and Animation. Since our 3D domain adaptation is designed to preserve the properties of $\mathcal{W}$ and $\mathcal{S}$ spaces, we can perform semantic edits via InterFaceGAN [51],

GANSpace [22], StyleSpace [62] *etc.*, and geometric edits using *TPS* (Sec. 3.4) and Δs interpolation (Sec. 3.2). To perform video editing, we design an encoder for EG3D based on *e4e* [58] to encode videos and transfer the edits from G_s to G_t based on the w codes [4, 6, 59]. We leave a more fine-grained approach for video processing as future work.

5. Results

5.1. Quantitative Results

In this section, we consider three important evaluations to verify the quality of the texture, geometry, and identity preservation in the new domain using the Caricature, Cartoons, and Pixar toons datasets. We evaluate the ablation of our design choices in the supplementary materials. In the evaluation, let G_{base} be the baseline naïve fine-tuning method which is trained with all the parameters using the losses in EG3D fine-tuned from FFHQ trained prior G_s. Note here we still align the cameras in G_{base} using the method defined in Sec. 3.1 and use adaptive discriminator [28] with $R1$ regularization for a fair comparison.

Texture Quality. To verify the quality of the texture, diversity of samples as well as to some extent, the geometry in the target domain T_t, we compare the *FID* [23] scores using G_{base} and G_t in Table 1. Note that in the case of Caricatures, we report two scores *i.e.* with and without using the attribute classifier loss in the training as discussed in Sec. 4. Notice that our method outperforms the naïve baseline method by a huge margin in some cases, especially in Caricatures and Cartoons. We attribute these differences to the mode collapse prone training of G_{base} which is correlated with flat geometry degenerate solution. We show visual results of the flat geometries learned by G_{base} and comparison in Fig. 2.

Geometric Quality. To quantify the flat geometries, in Table 2, we show three scores that help us understand such degenerate solutions. Here we consider coupled depth maps generated from sampling in the domains T_s (G_s) and T_t (G_t and G_{base}). First, we compute the expectation of the absolute mean differences (M_d) of the corresponding foreground depth maps sampled from T_s and T_t. We also compute the expectation of the absolute standard deviation differences (S_d) for the same setting. Here, we assume that the flatter geometries have a large difference in the depth maps as compared to the prior as indicated by M_d. Moreover, S_d computes the distance in the distribution of the depth values, where a larger difference indicates a narrow distribution, and hence a flatter geometry. We also notice that the flat geometry is correlated with the generator learning diverse poses when images are rendered under standard canonical camera parameters *i.e.* $M(0, 0, c, r)$. We hypothesize in the case of the flatter geometries, the model learns to pose in-

Figure 6. **Deformations using _TPS_.** Geometric edits using our proposed _TPS (Thin Plate Spline)_ module learned on the frontal tri-plane features. Each sub-figure shows a 3D avatar and three examples of _TPS_ deformations sampled from the learned 3D deformation space.

Table 1. **FID Computation.** FID (Fréchet Inception Distance) between the 2D dataset and the samples generated by the fine-tuned 3D GAN using baseline (G_{base}) and Ours (G_t). '*' represents the score with the inclusion of the attribute classifier loss discussed in Sec. 3.2.

Method	Caricatures	Cartoons	Pixar Toons
G_{base}	67.8	79.0	15.1
G_t (Ours)	**19.4/20.2***	**12.8**	**12.4**

Table 2. **Geometry Evaluation.** Comparing the geometry using baseline method (G_{base}) and Ours (G_t). For the definition of M_d, S_d and $R(T_2)$, refer to Sec. 5.1.

Metric	Method	Caricatures	Cartoons	Pixar
$M_d \downarrow$	G_{base}	0.47	0.21	0.29
	G_t (Ours)	**0.21**	**0.13**	**0.13**
$S_d \downarrow$	G_{base}	0.22	0.14	0.15
	G_t (Ours)	**0.15**	**0.10**	**0.09**
$R(T_2) \downarrow$	G_{base}	2.99	3.39	4.01
	G_t (Ours)	**2.27**	**1.62**	**1.56**

Table 3. **Identity Preservation.** Identity preservation using baseline (G_{base}) and Ours (G_t).

Method	Caricatures	Cartoons	Pixar Toons
G_{base}	1.28	0.92	0.85
G_t (Ours)	**0.87**	**0.81**	**0.73**

formation in the earlier layers instead of being camera view-dependent. To quantify this, since pose information may not be available for some domains (_e.g._ cartoons), we compute the $R(T_2)$ scores between corresponding images in the domain T_s (G_s) and T_t (G_t and G_{base}). Note that these scores are computed without the _TPS_ module. Our scores are lower in all three metrics, hence, validating that our method avoids the degenerate solution and preserves the geometric distribution of the prior. For discussion on the _TPS_ module and ablations refer to the supplementary materials.

Identity Preservation. Identity preservation score is another important evaluation to check the quality of latent space linking between G_s and G_t. In Table 3, we compute the attribute loss (_BCE_ loss) between the domains T_s and T_t using the attribute classifiers [24, 25]. Note that our method

Figure 7. **Local edits.** Local edits performed on the 3D avatars using the S space.

is able to preserve the identity better across the domains.

5.2. Qualitative Results

For qualitative results, we show the results of the domain adaptation, as well as the personalized edits (geometric and semantic), performed on the resultant 3D avatars. First, in order to show the quality of domain adaptation, identity preservation, and geometric consistency, in Fig. 3, we show results from G_s and corresponding results from 3D avatar generator G_t trained on Caricatures, Pixar toons, Cartoons, and Comic domains. Next, in order to show that the method generalizes to real images, we use the method described in
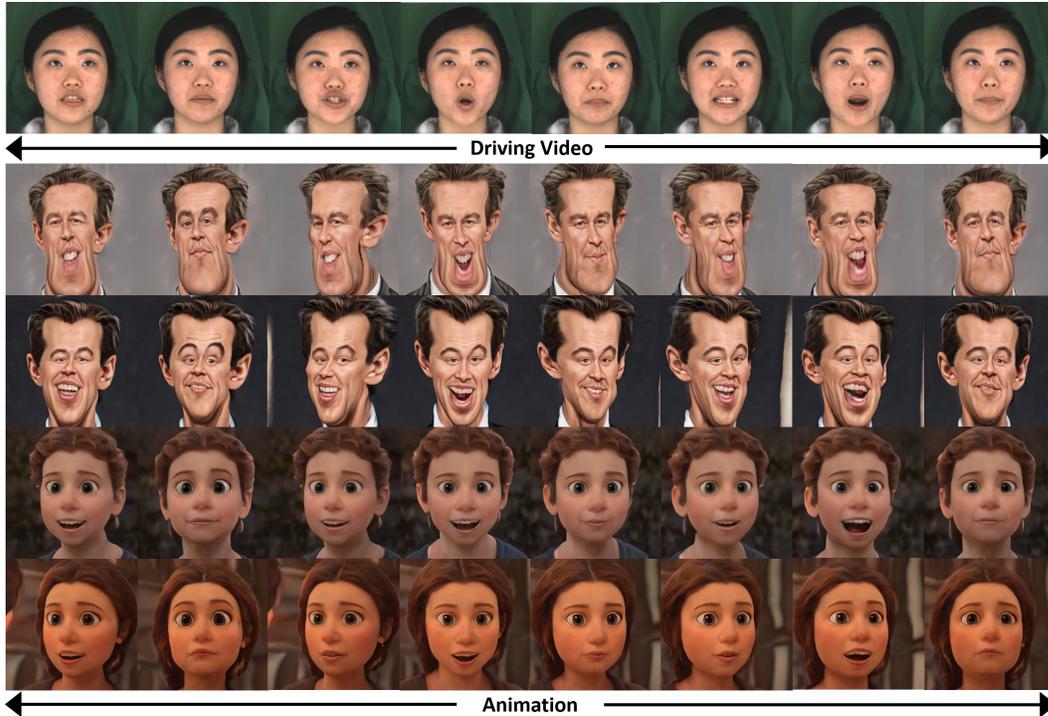
Figure 8. **3D avatar animation.** Animation of 3D avatars generated using a driving video encoded in source domain T_s and applied to samples in target domain T_t. The top row shows the driving video and the subsequent rows show generated animations using a random Caricature or Pixar toon. The head pose is changed in each frame of the generated animation to show 3D consistency.

Sec. 4 to project and transfer the latent code from G_s to G_t to produce the 3D avatars. In Fig. 4, we show our results of real to 3D avatar transfer. Notice the quality both in terms of texture as well as geometry for both these results achieved by our method. Next, we show geometric and semantic edits possible to produce personalized 3D avatars:

Geometry Edits. We show two type of geometric edits *i.e.* Δs interpolation (Sec. 3.2) and deformation using *TPS* (Sec. 3.4). First, in Fig. 5, we show the geometry interpolation by interpolating between original s activations of G_s and learned Δs parameters. In Fig. 6, we show some additional exaggerations in caricatures using the learned 3D deformation latent space of *TPS* module.

Semantic Edits and Animation. Since in our method, we encourage the latent regularization to preserve the properties of the latent space learned by the G_s generator, in Fig. 7 we show $\mathcal{S}$ space edits performed on the 3D avatars. Notice the quality of edits in terms of locality and adaptability. Additionally, we can edit semantics like hair as opposed to 3D morphable model based methods. In Fig. 8, thanks to the latent space semantics preservation ensured by our method, we can perform some video edits to create a coherent animation based on the difference of w codes of video encoded in G_s (Sec. 4) and applied to layers $7-10$ in G_t. Notice the quality of expressions, identity preservation, and 3D consistency across each identity in each row.

6. Conclusion

We tackled two open research problems in this paper. In the first part, we proposed the first domain adaptation method for 3D-GANs to the best of our knowledge. This part yields two linked EG3D generators, one in the photorealistic source domain of faces, and another EG3D generator in an artistic target domain. As possible target domains, we show results for cartoons, caricatures, and Comics. In the second part, we built on domain adaptation to create 3D avatars in an artistic domain that can be edited and animated. Our framework consists of multiple technical components introduced in this paper. First, we propose a technique for camera space estimation for artistic domains. Second, we introduce a set of regularizers and loss functions that can regularize the fine-tuning of EG3D in such a way that enough of the 3D structure and geometry of the original model is kept, while the distinguishing attributes of the artistic domain, such as textures and colors and local geometric deformations can still be learned. Third, we introduce a geometric deformation module that can reintroduce larger geometric deformations in a controlled manner. These larger geometric deformations can interact and cooperate with EG3D so that semantic edits are still possible. Finally, we propose an embedding algorithm that is especially suitable for two linked EG3D generator networks.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, Seoul, Korea, 2019. IEEE. 2, 6

[2] Rameen Abdal, Peihao Zhu, John Femiani, Niloy Mitra, and Peter Wonka. Clip2stylegan: Unsupervised extraction of stylegan edit directions. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery. 2

[3] Rameen Abdal, Peihao Zhu, Niloy J. Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Trans. Graph.*, 40(3), may 2021. 2, 3

[4] Rameen Abdal, Peihao Zhu, Niloy J. Mitra, and Peter Wonka. Video2stylegan: Disentangling local and global variations in a video, 2022. 6

[5] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 2

[6] Yuval Alaluf, Or Patashnik, Zongze Wu, Asif Zamir, Eli Shechtman, Dani Lischinski, and Daniel Cohen-Or. Third time's the charm? image and video editing with stylegan3. *CoRR*, abs/2201.13433, 2022. 6

[7] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit H. Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. *CoRR*, abs/2111.15666, 2021. 2

[8] Anonymous. 3d generation on imagenet. In *Open Review*, 2023. 3

[9] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[10] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021. 1, 2, 3

[11] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks, 2021. 2, 4

[12] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021. 2

[13] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. Inout: Diverse image outpainting via gan inversion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[14] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[15] Min Jin Chong and David A. Forsyth. Jojogan: One shot face stylization. *CoRR*, abs/2112.11641, 2021. 2

[16] Min Jin Chong, Hsin-Ying Lee, and David Forsyth. Stylegan of all trades: Image manipulation with only pretrained stylegan. *arXiv preprint arXiv:2111.01619*, 2021. 2

[17] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021. 2

[18] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators, 2021. 2

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[20] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2022. 2

[21] Fangzhou Han, Shuquan Ye, Mingming He, Menglei Chai, and Jing Liao. Exemplar-based 3d portrait stylization. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 2

[22] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*, 2020. 2, 6

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[24] Jing Huo, Wenbin Li, Yinghuan Shi, Yang Gao, and Hujun Yin. Webcaricature: a benchmark for caricature recognition. In *British Machine Vision Conference*, 2018. 3, 6, 7

[25] Wonjong Jang, Gwangjin Ju, Yucheol Jung, Jiaolong Yang, Xin Tong, and Seungyong Lee. Stylecarigan: Caricature generation via stylegan feature map modulation. 40(4), 2021. 2, 3, 6, 7

[26] Yucheol Jung, Wonjong Jang, Soongjin Kim, Jiaolong Yang, Xin Tong, and Seungyong Lee. Deep deformable 3d caricatures with learned shape control. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. ACM, aug 2022. 2

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. 2

[28] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020. 1, 2, 5, 6

[29] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020. 3

[30] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks, 2021. 1, 2

[31] 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*, pages 4401–4410, 2019. 1, 3

[32] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based generator architecture for generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4217–4228, Dec. 2021. 2

[33] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 2

[34] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 5

[35] Thomas Leimkühler and George Drettakis. Freestylegan: Free-view editable portrait rendering with the camera manifold. 40(6), 2021. 3

[36] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-pixel image synthesis. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[37] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 2

[38] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans, 2020. 2

[39] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. In *2021 International Conference on 3D Vision (3DV)*, pages 951–961. IEEE, 2021. 2

[40] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 2

[41] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. *arXiv preprint arXiv:2112.11427*, 2021. 1, 2

[42] Xingang Pan, Bo Dai, Ziwei Liu, Chen Change Loy, and Ping Luo. Do 2d gans know 3d shape? unsupervised 3d shape reconstruction from 2d image gans. *arXiv preprint arXiv:2011.00844*, 2020. 2

[43] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 2

[44] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021. 2

[45] Justin N. M. Pinkney and Doron Adler. Resolution dependent gan interpolation for controllable image synthesis between domains, 2020. 2, 3

[46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 2

[47] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. 2

[48] 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. *arXiv preprint arXiv:2008.00951*, 2020. 2

[49] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021. 2

[50] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[51] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2, 6

[52] Yichun Shi, Divyansh Aggarwal, and Anil K Jain. Lifting 2d stylegan for 3d-aware face generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6258–6266, 2021. 2

[53] Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 3d generation on imagenet. In *International Conference on Learning Representations (ICLR)*, 2023. 2

[54] Guoxian Song, Linjie Luo, Jing Liu, Wan-Chun Ma, Chunpong Lai, Chuanxia Zheng, and Tat-Jen Cham. Agilegan: Stylizing portraits by inversion-consistent transfer learning. *ACM Trans. Graph.*, 40(4), jul 2021. 2

[55] Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis, 2022. 1, 2

[56] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6142–6151, 2020. 2

[57] Ayush Tewari, Mohamed Elgharib, Mallikarjun BR, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt. Pie: Portrait image embedding for semantic control. volume 39, December 2020. 2

[58] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *arXiv preprint arXiv:2102.02766*, 2021. 2, 6

[59] Rotem Tzaban, Ron Mokady, Rinon Gal, Amit H. Bermano, and Daniel Cohen-Or. Stitch it in time: Gan-based facial editing of real videos. *CoRR*, abs/2201.08361, 2022. 6

[60] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Cross-domain and disentangled face manipulation with 3d guidance. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 2

[61] WarBean. tps-stn-pytorch. https://github.com/WarBean/tps_stn_pytorch. 5

[62] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. *arXiv preprint arXiv:2011.12799*, 2020. 2, 4, 6

[63] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, et al. Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 2

[64] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. *arXiv preprint arXiv:2112.10759*, 2021. 1, 2

[65] Shuai Yang, Liming Jiang, Ziwei Liu, and Chen Change Loy. Pastiche master: Exemplar-based high-resolution portrait style transfer. In *CVPR*, 2022. 2, 3

[66] Zipeng Ye, Mengfei Xia, Yanan Sun, Ran Yi, Minjing Yu, Juyong Zhang, Yu-Kun Lai, and Yong-Jin Liu. 3d-CariGAN: An end-to-end solution to 3d caricature generation from normal face photos. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2021. 2

[67] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 2

[68] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[69] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European Conference on Computer Vision*, pages 592–608. Springer, 2020. 2

[70] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Mind the gap: Domain gap control for single shot domain adaptation for generative adversarial networks. In *International Conference on Learning Representations*, 2022. 2

[71] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents?, 2020. 2

[72] zllrunning. face-parsing.pytorch. https://github.com/zllrunning/face-parsing.PyTorch. 2, 3