# VToonify: Controllable High-Resolution Portrait Video Style Transfer

SHUAI YANG, S-Lab, Nanyang Technological University, Singapore
LIMING JIANG, S-Lab, Nanyang Technological University, Singapore
ZIWEI LIU, S-Lab, Nanyang Technological University, Singapore
CHEN CHANGE LOY*, S-Lab, Nanyang Technological University, Singapore


Fig. 1. **Controllable High-Resolution Portrait Video Style Transfer.** We propose a novel VToonify framework that can synthesize high-resolution stylized videos from a low-resolution input. VToonify precisely pastiches the facial structure style of style collections of Caricature and Cartoon. Within each style collection, fine-level style transfer in terms of both structure style and color style is supported by specifying a reference style image. Besides style types, our framework further supports the adjustment of style degrees to flexibly choose to which degree the original facial features are preserved. Input video: ©Pexels Ketut Subiyanto.

Generating high-quality artistic portrait videos is an important and desirable task in computer graphics and vision. Although a series of successful portrait image toonification models built upon the powerful StyleGAN have been proposed, these image-oriented methods have obvious limitations when applied to videos, such as the fixed frame size, the requirement of face alignment, missing non-facial details and temporal inconsistency. In this work, we investigate the challenging controllable high-resolution portrait video style transfer by introducing a novel **VToonify** framework. Specifically, VToonify leverages the mid- and high-resolution layers of StyleGAN to render high-quality artistic portraits based on the multi-scale content features extracted by an encoder to better preserve the frame details. The resulting fully convolutional architecture accepts non-aligned faces in videos of variable size as input, contributing to complete face regions with natural motions in the output. Our framework is compatible with existing StyleGAN-based image toonification models to extend them to video toonification, and inherits appealing features of these models for flexible style control on color and intensity. This work presents two instantiations of VToonify built upon Toonify and DualStyleGAN for collection-based and exemplar-based portrait video style transfer, respectively. Extensive experimental results demonstrate the effectiveness of our proposed VToonify framework over existing methods in generating high-quality and temporally-coherent artistic portrait videos with flexible style controls. Code and pretrained models are available at our project page: www.mmlab-ntu.com/project/vtoonify/.

*Corresponding Author

Authors' addresses: Shuai Yang, S-Lab, Nanyang Technological University, Singapore, shuai.yang@ntu.edu.sg; Liming Jiang, S-Lab, Nanyang Technological University, Singapore, liming002@ntu.edu.sg; Ziwei Liu, S-Lab, Nanyang Technological University, Singapore, ziwei.liu@ntu.edu.sg; Chen Change Loy, S-Lab, Nanyang Technological University, Singapore, ccloy@ntu.edu.sg.

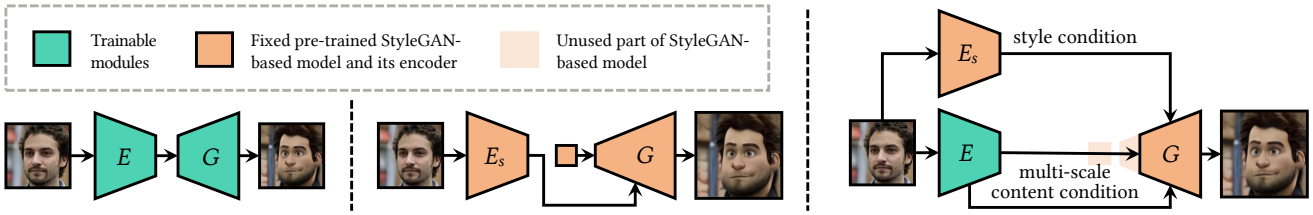
CCS Concepts: • **Computing methodologies** → **Computational photography**; *Computer vision.*

Additional Key Words and Phrases: Face toonification, exemplar-based, model distillation, StyleGAN

## 1 INTRODUCTION

Artistic portraits are ubiquitous in our daily life as well as creative industries in forms of arts, social media avatars, movies, entertainment advertising, *etc.* With the advent of deep learning technology, one can now render high-quality artistic portraits from real face images through automatic portrait style transfer. There are a number of successful approaches designed for image-based style transfer [Alaluf et al. 2021; Gatys et al. 2016; Jang et al. 2021; Pinkney and Adler 2020; Selim et al. 2016; Song et al. 2021; Yang et al. 2022], with many of them easily accessible to novice users in the form of mobile applications. Over the last few years, video content has quickly become a staple of our social media feeds. The popularity of social media and ephemeral videos arouse increasing demands for creative editing of videos, such as portrait video style transfer, to create effective and engaging videos. Existing image-oriented approaches have several drawbacks when applied to videos, limiting their applications in automatic portrait video stylization.

The ability to generate high-quality 1024×1024 faces with flexible style control makes StyleGAN a popular backbone for building a portrait image style transfer model. Such a StyleGAN-based framework (also called image toonification) mainly encodes a real face into the StyleGAN latent space, and applies the resulting style code




(a) Image translation framework (b) StyleGAN-based framework (c) The proposed hybrid framework

Fig. 2. **Overview of the proposed hybrid framework.** (a) Image translation framework uses fully convolutional networks to support variable input size. However, training from scratch renders it difficult for high-resolution and controllable style transfer. (b) The StyleGAN-based framework leverages the pre-trained model for high-resolution and controllable style transfer, but is limited to fixed image size and detail losses. (c) Our hybrid framework combines the above two ideas, supporting high-resolution and controllable style transfer on videos of various sizes.

to another StyleGAN fine-tuned on the artistic portrait dataset to obtain its stylized version [Pinkney and Adler 2020]. A key problem with this framework is that StyleGAN produces images with aligned faces and under a fixed size, which does not favor dynamic faces in real videos like Fig. 1. Cropping and aligning faces in the video often results in an incomplete face and unnatural motions. We refer to this problem as the 'fixed-crop limitation' of StyleGAN. Although StyleGAN3 [Karras et al. 2021] has been proposed for unaligned faces, it only supports a fixed image size. Moreover, compared to aligned faces, a recent study [Alaluf et al. 2022a] finds that it becomes more difficult to encode unaligned faces. An inaccurate face encoding is detrimental to portrait style transfer, causing problems such as identity change and missing items in the reconstructed frames and the stylized frames.

Different from the StyleGAN-based framework, image-to-image translation directly learns to map the input image from the real face domain to the artistic portrait domain. This problem is often tackled using a fully convolutional encoder-generator architecture, eschewing strict restrictions on the image size and the face positions during the test phase. Due to the lack of paired training data, the framework typically builds upon the cycle consistency paradigm [Zhu et al. 2017] to learn bi-directional mappings. However, the complicated mappings constrain the models to a small image size of 256×256. There are also frameworks [Viazovetskyi et al. 2020] distilling StyleGAN for face attribute editing, where the StyleGAN is used to generate paired data for training. The strong paired supervision enables the framework to handle high-resolution images. Nevertheless, the above method itself does not incorporate StyleGAN into its network design. Thus, StyleGAN's good properties of flexible style control are unfortunately not inherited. Recently, GLEAN [Chan et al. 2021] introduces StyleGAN to the translation network design to effectively use the StyleGAN face priors but entails the accompanying 'fixed-crop limitation'. Despite the limitation, this hybrid framework shows high performance and inspires our work.

As analyzed above, an effective method needs to address the following challenges for portrait video style transfer: **1)** The method needs to cope with unaligned faces and different video sizes to maintain natural motions. Large video size, or wide angle of view, can capture more information and prevent the face from moving out of the frame. **2)** Generating high-resolution videos is desired to match the widely used HD devices nowadays. **3)** To build a practical user interaction system, flexible style control should be provided for users to adjust and select their preference.

To this end, we propose a new hybrid framework **VToonify** for video toonification. We first analyze the translation equivariance in StyleGAN which constitutes our key solution to overcome the fixed-crop limitation. As shown in Fig. 2(c), VToonify combines the merits of the StyleGAN-based framework [Pinkney and Adler 2020] and the image translation framework [Viazovetskyi et al. 2020], achieving controllable high-resolution portrait video style transfer. We adopt the StyleGAN architecture following [Pinkney and Adler 2020] for high-resolution style transfer, but adapt StyleGAN by removing its fixed-sized input feature and low-resolution layers to construct a novel fully convolutional encoder-generator architecture akin to that in the image translation framework, which supports different video sizes. Apart from the original high-level style code, we train an encoder to extract multi-scale content features of the input frame as the additional content condition to the generator, so that the key visual information of the frame can be better preserved during style transfer. We follow [Chen et al. 2019; Viazovetskyi et al. 2020] to distill StyleGAN on its synthesized paired data. We further propose a flicker suppression loss based on the simulation of camera motion over a single synthetic data to eliminate flickers. Therefore, VToonify is able to learn a fast and coherent video translation without real data, complicated video synthesis or explicit optical flow computation. Different from the standard image translation framework in [Chen et al. 2019; Viazovetskyi et al. 2020], VToonify incorporates the StyleGAN model into the generator to distill both data and model. Therefore, VToonify inherits the style adjustment flexibility of StyleGAN. By reusing StyleGAN as the generator, we only need to train the encoder, greatly reducing both the training time and training difficulty.

With the novel formulation above, this paper presents two variants of VToonify built upon two representative StyleGAN backbones, Toonify [Pinkney and Adler 2020] and DualStyleGAN [Yang et al. 2022], for collection-based and exemplar-based portrait video toonification, respectively. The former stylizes faces based on the overall style of the dataset, while the later uses a single image in the dataset to specify finer-level style as shown on the top right of Fig. 1. By adapting DualStyleGAN's style control modules [Yang et al. 2022] to adjust the features of our encoder and elaborately

designing the data generation and training objectives, VToonify inherits DualStyleGAN's flexible style control and adjustment of style degrees, and further extends these features to videos (*e.g.*, the top right of Fig. 1). We show in the experiment that VToonify generates stylized frames not only as high-quality as the backbones but also better preserving the details of the input frame. To summarize, our main contributions are as follows:

- We analyze the fixed-crop limitation of StyleGAN and suggest a corresponding solution based on the translation equivariance in StyleGAN.
- We propose a novel fully convolutional VToonify framework for controllable high-resolution portrait video style transfer, supporting unaligned faces and various video sizes.
- We build VToonify upon Toonify and DualStyleGAN backbones and distill the backbones in terms of both data and model, to realize collection-based and exemplar-based portrait video style transfer.
- We design a principled data-friendly training scheme and propose an optical-flow-free flicker suppression loss for temporal consistency, which is efficient and effective for training a video style transfer model.

## 2 RELATED WORK

### 2.1 Face Image Style Transfer

*Image-to-Image Translation.* To learn the structure style, image translation framework is adopted for face style transfer. Pix2pix [Isola et al. 2017] first proposes the supervised image translation framework to map images from a source domain to a target domain. This framework requires paired data for training, which is not easy to collect for face style transfer. To solve this problem, some methods like AnimeGAN2 [Chen et al. 2019] distill existing style transfer models [Pinkney and Adler 2020] by generating synthetic face-cartoon pairs for training. Meanwhile, CariGAN [Cao et al. 2018] solves a weakly supervised task where the paired face and caricature share the same identity but without pixel-level correspondences.

To support unpaired data, cycle consistency is proposed in CycleGAN [Zhu et al. 2017] for unsupervised image translation. The following face-to-cartoon translation methods improve CycleGAN by incorporating attention to the domain-specific features [Kim et al. 2019], data augmentation to strengthen domain invariance [Chong and Forsyth 2021], and matching multi-scale statistical features to stabilize training [Shao and Zhang 2021]. Cycle consistency sometimes can be restrictive for imbalanced domains, *e.g.*, transforming abstracted anime face back to real faces. To solve this problem, CouncilGAN [Nizan and Tal 2020] leverages collaboration between multiple generators instead of cycle consistency while AniGAN [Li et al. 2021] constrains translations with multi-scale style losses. One main drawback of the unpaired image translation framework is that learning from scratch on complicated translations makes this framework limited to low-resolution images.

By comparison, besides distilling data, we further incorporate the StyleGAN model into our network design, thus supporting both high-resolution video generation and flexible style control.

*StyleGAN-Based Toonification.* StyleGAN [Karras et al. 2019, 2020b] generates photo-realistic face images and provides hierarchical style control, serving as a powerful backbone for face stylization. Toonify [Pinkney and Adler 2020] fine-tunes a pre-trained StyleGAN on a cartoon dataset and combines the shallow layers of the fine-tuned model with the deep layers of the original model to generate faces in cartoon structures while maintaining the realistic facial color and textures. The pSp method [Richardson et al. 2021] accelerates Toonify by training an encoder to project real face images into the closest cartoon faces in the fine-tuned latent space. AgileGAN [Song et al. 2021] and ReStyle [Alaluf et al. 2021] improve pSp by a variational encoder and an iterative refinement mechanism, respectively. StyleCariGAN [Jang et al. 2021] combines StyleGAN with image translation by explicitly learning structure transfer with cycle translations. DualStyleGAN [Yang et al. 2022] extends StyleGAN with an extrinsic style path to accept the condition from a style image for exemplar-based style transfer. StyleGAN-NADA [Gal et al. 2022] proposes to shift StyleGAN to new artistic domains guided by CLIP [Radford et al. 2021] without using any real cartoon dataset, realizing text-driven toonification.

Although high-quality results are shown, the above methods assume a face is well aligned, which is unrealistic for videos. By comparison, VToonify supports end-to-end unaligned face stylization in videos of variable size.

### 2.2 StyleGAN Inversion

StyleGAN inversion aims to project real face images into the latent space of StyleGAN for editing. Image2StyleGAN [Abdal et al. 2019] proposes to project real images into the $\mathcal{W}+$ space through optimization. PSp [Richardson et al. 2021] trains an encoder to accelerate the projection. E4e [Tov et al. 2021] predicts the latent codes in $\mathcal{W}+$ that reside close to the $\mathcal{W}$ space to improve the editability. Instead of searching the optimal latent code, PTI [Roich et al. 2022] fine-tunes StyleGAN to approach the target image based on its predicted $\mathcal{W}$-space latent code. Based on this idea, HyperInverter [Dinh et al. 2022] and HyperStyle [Alaluf et al. 2022b] propose to train a hyper network to directly predict the offsets of the StyleGAN parameters to simulate the fine-tuning in a more efficient way. Recently, methods [Parmar et al. 2022; Wang et al. 2022; Zhu et al. 2021a] have been proposed to invert real images into the feature space ($\mathcal{F}$ space) of StyleGAN, which could better reconstruct the image details and is suitable for spatial editing. Our encoder also extracts content features in addition to the latent code as the condition to StyleGAN, which can be seen as a hybrid $\mathcal{F}$-$\mathcal{W}+$ space. Different from the above methods, our method uses the $\mathcal{F}$ space to overcome StyleGAN's fixed-crop limitation. We refer to [Xia et al. 2022] for a comprehensive survey on StyleGAN inversion.

### 2.3 Video Style Transfer

*Optical-Flow-Based Method.* Video style transfer pays additional attentions to the temporal consistency, which is usually achieved by optical flows. [Ruder et al. 2016] and [Gupta et al. 2017] apply Neural Style Transfer [Gatys et al. 2016] to videos by using the optical flow of the input video to warp the previous stylized frame to the current frame for frame initialization or extra conditions. [Huang et al. 2017]

and [Chen et al. 2017] use a temporal consistency loss to constrain the coherence between corresponding pixels of consecutive stylized frames. [Wang et al. 2020] propose a compound regularization to better fit the nature of temporal variation with both optical flows and local jitters. The above approaches assume identical optical flows between the input frames and between the stylized frames, which does not hold for toonification where the faces are often deformed. Unlike previous studies, we formulate an effective and simple-to-implement flicker suppression loss by simulating the motion of camera over a single frame.

*Image-Animation-Based Method.* Another way of synthesizing artistic portrait videos is to apply image animation methods like First Order Motion [Siarohin et al. 2019] to animate an artistic portrait with the motion driven by the video. Recently, DaGAN [Hong et al. 2022] shows promising results by estimating the dense 3D geometry from the face videos to constrain the output to be more consistent with 3D face structures. Such methods, however, only use the information of a single stylized frame, inevitably losing important details. As we will show later, even the 3D geometry is considered, the motion of the facial textureless region is still hard to predict, leading to annoying flickers.

*StyleGAN-Based Video Editing.* Besides image editing [Härkönen et al. 2020; Jiang et al. 2021; Shen et al. 2020; Shen and Zhou 2021], StyleGAN has also gained attention in video editing. [Fox et al. 2021] project video frames into the sequences of StyleGAN latent codes, and train a network to capture the temporal correlations from these low-dimensional codes. [Yao et al. 2021] train a latent transformation network that disentangles the identity and facial attributes to edit the face with better identity preservation. [Liu et al. 2022] design a sketch branch combined with StyleGAN for sketch-based video editing. To seamlessly stitch the cropped edited face and the background, STIT [Tzaban et al. 2022] proposes to tune StyleGAN to provide spatially-consistent transitions. All the above methods require face alignment and cropping as pre-processing. While StyleGAN3 [Karras et al. 2021] is proposed to support non-aligned faces, recent study [Alaluf et al. 2022a] shows that such pre-processing is still needed for valid encoding in StyleGAN3, followed by projecting the results back to the original context. Results generated through such a complicated process are prone to artifacts.

By comparison, VToonify is an end-to-end framework without the above data pre-processing and post-processing.

## 3 ANALYSIS OF STYLEGAN-BASED TOONIFICATION

Before we start introducing VToonify, we first briefly look into the problem of StyleGAN-based toonification when applied to video. We will analyze the causes of the problem and present our corresponding solutions. Here we focus on StyleGAN/StyleGAN2 [Karras et al. 2019, 2020b], since most existing StyleGAN-based toonification models are built upon them.

*Can a pre-trained StyleGAN generate unaligned faces?* The translation equivariance of convolutional networks naturally supports face translation. As shown in Fig. 3(b), we first obtain the 32×32 feature from the 7-th layer of StyleGAN, and apply translation to it with reflection padding. The translated feature is decoded by the last

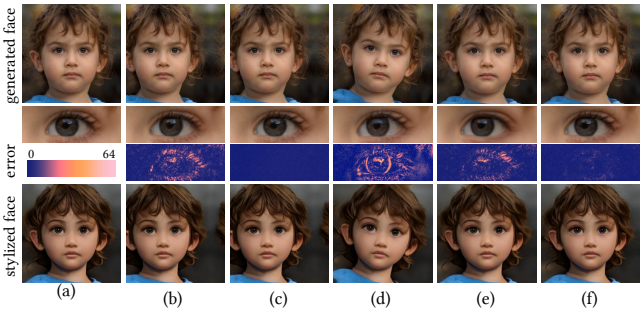



Fig. 3. **Analysis of the StyleGAN** in generating alias free unaligned faces. (a) Generated face by StyleGAN. (b) Face generated by translating the feature map of the StyleGAN 7-th layer by 128 pixels. Compared to (b), (c) additionally translates noise maps after the 7-th layer. (d) Face generated by rotating the feature map of the StyleGAN 7-th layer 10 degrees. (e) Face generated by removing the noise inputs after the 7-th layer. (f) Face generated by removing the noise inputs after the 13-th layer. Each groups show the generated face, the enlarged (re-aligned) right eyes, the error maps with (a), and the toonified version by DualStyleGAN.

11 layers of StyleGAN, which as expected is exactly the translated version of the original face. We further try feature rotations and find that the output faces are rotated and look still plausible within a small rotation ranges in Fig. 3(d). This implies that the last 11 layers of StyleGAN, which mainly render colors and textures for faces, is robust to geometric transformations and could be used for high-resolution unaligned face generation.

*Can a pre-trained StyleGAN generate images under different sizes?* The fully convolutional architecture naturally supports different input sizes since the convolution operation is independent of input size. The reason why StyleGAN is limited to a fixed size is its fixed-sized input feature as shown in Fig. 2(b). By removing this fixed input and adding another encoder for providing features of variable size, we can build a fully convolutional encoder-generator architecture as in neural style transfer [Johnson et al. 2016] and image translation [Zhu et al. 2017] for different input sizes. Moreover, the encoder directly extracts 32×32 features from unaligned faces instead of artificially aligning faces and transforming their features.

*Where do flickers come from?* In our experiment on translating the 32×32 feature, we notice flickers in the output. If we compare the translated face with the original one, we can observe the changes in textures as visualized by the error map in Fig. 3(b). The cause is the additional noise inputs to enrich texture details. Random noises create random texture details while a fixed noise leads to 'texture sticking' artifacts [Karras et al. 2021]. Removing the noise inputs can solve this problem but at cost of missing details as in Fig. 3(e). The problem can also be solved by translating the noise maps in the last 11 layers along with the 32×32 feature (Fig. 3(c)). However, the motion in real applications is more complex and the face deformation during style transfer makes the estimation of such motion in each layer impractical. Our solution is to remove the noise inputs and to pass the multi-scale encoder features to the generator to supplement the texture details, as illustrated in Fig. 2(c). We

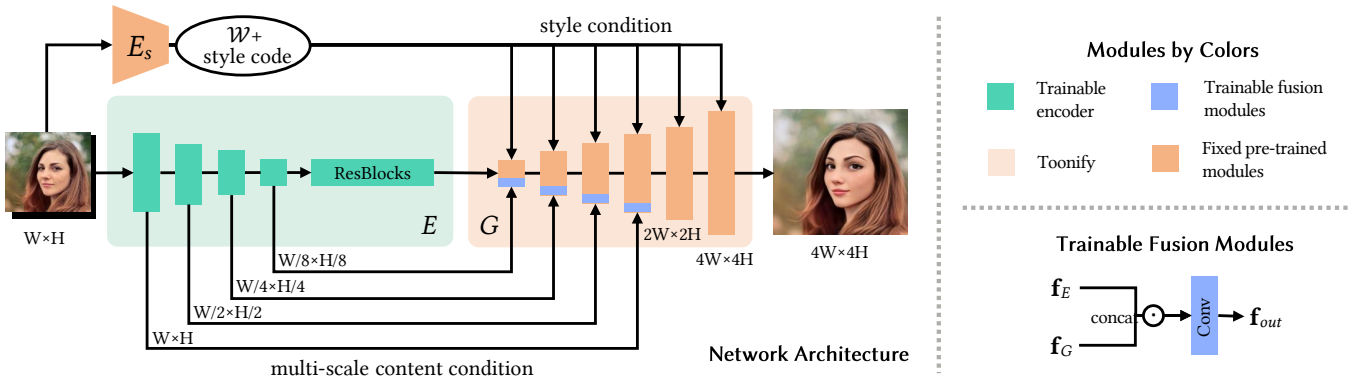



Fig. 4. **Framework of the proposed collection-based portrait video style transfer.** We use the highest 11 layers of the pre-trained Toonify as our generator $G$. In addition to the style code as condition, $G$ is also conditioned on the multi-scale content features from the encoder $E$. In the training phase, we use 256×256 images as input, while during testing, our fully convolutional network accepts frames of various sizes. Input video: ©Pexels Andrea Piacquadio.

further find that the noise inputs to the high-resolution layers have little effect as in Fig. 3(f), meaning high-resolution encoder features are not necessary. Thus, we can safely follow the common practices to use low-resolution videos as input to generate high-resolution videos, which improves efficiency and reduces the requirement for input resolution.

## 4 VTOONIFY FOR VIDEO TOONIFICATION

### 4.1 Collection-Based Portrait Video Style Transfer

We start from a simple case: collection-based portrait video style transfer. The collection-based task transfers a holistic style of the style collection. We leverage the representative Toonify [Pinkney and Adler 2020] as the backbone, which uses the original StyleGAN architecture and is solely conditioned on the style code.

*4.1.1 Network Architecture.* As shown in Fig. 4, the collection-based VToonify framework contains an encoder $E$ and a generator $G$ built upon Toonify. $E$ accepts video frames and generates content features, which are then fed into $G$ to produce the final stylized portraits. Unlike existing StyleGAN-based frameworks that use the whole StyleGAN architecture, we only use the highest 11 layers of StyleGAN to build our $G$. As analyzed in [Karras et al. 2019], the low-resolution and high-resolution layers of StyleGAN mainly capture structure-related styles and color/texture styles, respectively. Therefore, the main task of $G$ is to upsample the content features and render stylish colors and textures for them.

Following the common practices of StyleGAN-based framework, we incorporate a StyleGAN encoder $E_s$ to provide $G$ with its style condition. Specifically, $E_s$ embeds the input frame into the StyleGAN $\mathcal{W}+$ latent space with eighteen 512-dimensional vectors, serving as the inputs to the eighteen layers of StyleGAN. We use the correspondingly last 11 vectors as the style condition of $G$.

Besides the style condition, $E$ downsamples the input $W \times H$ frame and extract its $W/8 \times H/8$ content feature with downsampling convolutional layers and ResBlocks [He et al. 2016]. The content feature replaces the original fixed-sized input features of StyleGAN in the 8-th layer, so that variable frame size is allowed in our framework (as long as $W$ and $H$ are divisible by 8).

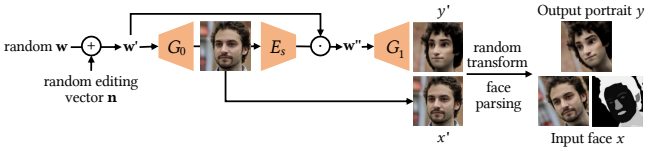



Fig. 5. **Pipeline of paired training data generation** for training collection-based portrait style transfer.

As analyzed in Sec. 3, the application of noises enriches the texture details but creates flickers on videos. To solve this problem, we remove the noise input and pass the mid-layer features of $E$ with rich low-level information to the corresponding resolution layers of $G$. Within each layer of $G$, we add a new fusion module to combine the generator feature $\mathbf{f}_G$ and the passed encoder feature $\mathbf{f}_E$. As shown in the bottom right of Fig. 4, $\mathbf{f}_G$ and $\mathbf{f}_E$ are concatenated channel-wisely as $\mathbf{f}_G \odot \mathbf{f}_E$ and then go through a convolutional layer to obtain the fused feature $\mathbf{f}_{out}$ as the input of the generator's next layer. The mid-layer features and the last-layer content feature form multi-scale content condition for $G$, effectively preserving the image details of the input frame compared to only using the style condition in previous StyleGAN-based approaches.

In summary, we remove the fixed-sized low-resolution layers of StyleGAN to support variable input size, and replace the noise inputs with multi-scale content conditions to prevent flickers and preserve the input frame details.

*4.1.2 Data Generation and Training Objectives.*

*Data Generation.* To distill Toonify for high-resolution toonification, we generate paired data based on it to train our model. The data generation process is intuitively illustrated in Fig. 5. Here, we use the original StyleGAN $G_0$ trained on real human faces and the fine-tuned StyleGAN $G_1$ in Toonify ($G$ is the highest 11 layers of $G_1$) to synthesize the 1024×1024 face image $x$ and its corresponding 1024×1024 ground truth stylized portrait $y$, respectively.

Specifically, we first sample random style codes $\mathbf{w} \in \mathcal{W}+$ to synthesize face images with $G_0$. StyleGAN tends to generate smiling faces with eyes open and mouths half-opened, introducing significant bias into the data. To solve this problem, we leverage the editing vectors $\mathbf{n}$ obtained from [Zhu et al. 2021b] to explicitly edit these face attributes. The vectors $\mathbf{n}$ are randomly sampled from {do nothing, remove smile, close eyes, close mouth, open mouth} and added to $\mathbf{w}$ to obtain the edited style code $\mathbf{w}'$. The corresponding face with edited attribute is then synthesized as $x' = G_0(\mathbf{w}')$. To obtain its stylized counterpart, we further compute its embedded style code $E_s(x'_\downarrow)$ ($E_s$ requires 256×256 inputs so we downsample $x'$ to $x'_\downarrow$ by 4×) and generate a new style code $\mathbf{w}''$ by concatenating the first 7 vectors of $\mathbf{w}'$ and the last 11 vectors of $E_s(x'_\downarrow)$. And the stylized portrait is obtained by $y' = G_1(\mathbf{w}'')$. Note that in Toonify, stylized portraits are normally obtained as $G_1(\mathbf{w}')$ for synthetic face inputs or $G_1(E_s(x'_\downarrow))$ for real face inputs. By comparison, our $y'$ uses the structure style part of $\mathbf{w}'$ to maintain better identity of $x'$, and the color and texture style part of $E_s(x'_\downarrow)$ to ensure its consistency with the form of style condition in our framework (see Fig. 4).

Finally, to make our model learn to handle unaligned faces, we augment $x'$ and $y'$ with random geometric transformations [Karras et al. 2020a] like flipping, scaling, translation and rotation. The resulting $\{x, y\}$ together with the style code $\mathbf{w}''$ form the paired training data of our method. To help the network better identify the face regions of $x$ especially at the beginning of training, we obtain its face parsing map by applying BiSeNet [Yu et al. 2018] to $x$, which serves as the additional semantic channels of $x$ in our framework (*i.e.*, the black channels concatenated to the input frame in Fig. 4). For simplicity, we will use $x$ to denote the concatenation of the face image and its parsing map in the following.

*Model Initialization.* Different from the well-defined $G$, $E$ is randomly initialized. To stabilize the training, we first pre-train $E$. Since the last-layer feature of $E$ replaces the original 8-th-layer input feature of $G_1$, we would like these two features match. Let $\mathbf{f}_E^{(\text{last})}(x)$ denote the last-layer feature of $E$ given a frame $x$ and $\mathbf{f}_{G_1}^{(8)}(\mathbf{w})$ denote the 8-th-layer input feature of $G_1$ given a style code $\mathbf{w}$. We pre-train $E$ to minimize the objective:

$$\mathcal{L}_{\text{E}} = \|\mathbf{f}_E^{(\text{last})}(x'_\downarrow) - \mathbf{f}_{G_1}^{(8)}(\mathbf{w}'')\|_2. \tag{1}$$

Besides, we initialize the fusion modules to map $\mathbf{f}_G \odot \mathbf{f}_E$ to $\mathbf{f}_G$. This is realized by setting the half part of the convolution kernel that interacts with $\mathbf{f}_G$ to the identity matrix and the other half part that interacts with $\mathbf{f}_E$ to values close to 0. With the above pre-trained encoder and initialization, our model exactly mimics the function of $G_1$ without noise inputs and could generate a rough stylized portrait as shown in Fig. 6(c). Up to this point, our model is ready for the subsequent training to generate facial details and support unaligned faces as in Fig. 6(d).

*Training Objectives.* We would like to stylize the low-resolution $x_\downarrow$ (4× downsampling of $x$) to approach the high-resolution ground truth $y$, leading to a reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \lambda_{\text{mse}}\|\widetilde{y} - y\|_2 + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}}(\widetilde{y}, y), \tag{2}$$

(a) input (b) ground truth (c) pretrain $E$ (d) full training

Fig. 6. **Training target and results of each stage.** (a) and (b) are the synthetic paired data for training VToonify. Bottom row applies geometric transformations. (c) Results of VToonify after pre-training $E$ and initializing the fusion modules. (d) Results of VToonify after full training.

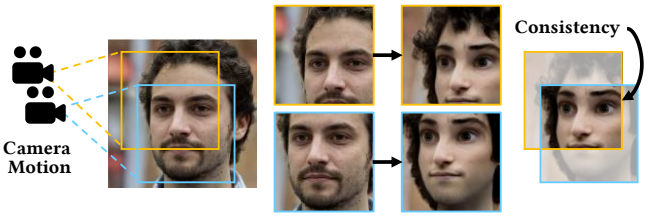



Fig. 7. **Illustration of the proposed flicker suppression** loss.

where $\widetilde{y} = G(E(x_\downarrow), \mathbf{w}'')$ is the stylization result given the content condition $E(x_\downarrow)$ and the style condition $\mathbf{w}''$. $\mathcal{L}_{\text{perc}}(\widetilde{y}, y)$ is the perceptual loss [Johnson et al. 2016] to minimize the semantic difference between $\widetilde{y}$ and $y$. Hyper-parameters $\lambda$s are used to balance different loss terms. The realism of $\widetilde{y}$ is further reinforced through an adversarial training with a discriminator $D$,

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_y[\log D(y)] + \mathbb{E}_x[\log(1 - D(\widetilde{y}))]. \tag{3}$$

Temporal consistency should be specially considered in our video style transfer task. It is impractical to generate paired video data with accurate optical flows and high temporal consistency from image-oriented Toonify. Instead, we propose to simulate the motion of camera over a single frame. As illustrated in Fig. 7, the parallel movement of the camera with a limited field of view captures two subframes from a single frame. If the camera is sufficiently far away from the face, one can ignore the 3D effects and the overlapped region of these two subframes should be the same. Correspondingly, the overlapped region of their stylized results should be kept the same. This idea leads to our flicker suppression loss:

$$\mathcal{L}_{\text{tmp}} = \|f_c(\widetilde{y}) - G(E(f_c(x)_\downarrow), \mathbf{w}'')\|_2, \tag{4}$$

where $f_c(\cdot)$ is the random cropping operation and we set one of the subframes as the whole frame for simple computation. Here, the frame $x$ is cropped in the original resolution before downsampling so that sub-pixel motion (thus temporal consistency) is supported. Although this loss does not aim to explicitly maintain temporal coherency and does not explicitly calculate optical flow, it intrinsically assumes a uniform optical flow. We find this simple solution

(a) $I$ (b) visualized $p$ (c) result with $p$ (d) visualized $\hat{p}$ (e) result with $\hat{p}$

Fig. 8. **Effect of face parsing map smoothing.** The proposed face parsing map smoothing algorithm relieves the inconsistency between adjacent parsing maps in (b). The resulting smoothed parsing maps in (d) help synthesize more consistent results as in (e). The local regions are enlarged with their contrast enhanced for better visual comparison. In (b)(d), we visualize the parsing map by transforming it to the discrete label map. Input video: ©Pexels Mikael Blomkvist.

effective to relieve flickers. Our full objective takes the form of

$$\min_G \max_D \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{tmp}}\mathcal{L}_{\text{tmp}}. \tag{5}$$

*Face Parsing Map Smoothing.* The temporal consistency of the stylized frames rely on the assumption that the input frames are temporally consistent, which is true for real video inputs. However, in our framework, the video frame is augmented with its parsing map. The inconsistency in parsing maps might violate this assumption. To solve this issue, we propose a face parsing map smoothing algorithm to strengthen the temporal consistency between consecutive face parsing maps. First, instead of using the discrete label map as the parsing map, we use the continuous probability map before the final argmax operation of BiSeNet [Yu et al. 2018]. Second, we may apply an optical-flow based frame fusion to the parsing maps. Specifically, we calculate the optical flow between the frames within a temporal window. Let $I_i$ and $p_i$ denote the $i$-th frame and its parsing map, respectively, and $k$ denote the temporal window size. The operation to warp $I_j$ to $I_i$ with their optical flow $f_{j,i}$ estimated by RAFT [Teed and Deng 2020] is defined as $\theta(I_j, f_{j,i})$. The corresponding occlusion mask is $m_{j,i}$. Then, the parsing map $p_i$ is fused and updated with its surrounding $2k+1$ parsing maps with temporal-spatial weights:

$$\hat{p}_i = \sum_{j\in[i-k,i+k]} \frac{w_{j,i} \otimes \theta(p_j, f_{j,i})}{w_i} \tag{6}$$

$$w_{j,i} = exp\Big(-\frac{(i-j)^2}{2\sigma_t^2} - \frac{\|I_i - \theta(I_j, f_{j,i})\|^2}{2\sigma_s^2}\Big) \otimes m_{j,i} \tag{7}$$

where $w_i = \sum_{j\in[i-k,i+k]} w_{j,i}$ is the denominator for normalization and $\otimes$ is the element-wise multiplication. Intuitively, the better matched regions in more adjacent frames have greater fusion weights. As shown in Fig. 8, the flickers in the brick shadow are effectively smoothed by considering the adjacent predictions, and the brick is more consistently stylized.

## 4.2 Exemplar-Based Portrait Video Style Transfer

We move to a more complex case: exemplar-based portrait video style transfer. This task stylizes human faces based on the style of a reference style image. We use DualStyleGAN [Yang et al. 2022] as the backbone, which adds an extrinsic style path to StyleGAN, and is conditioned on the intrinsic style code, the extrinsic style code and the style degree. The intrinsic style code depicts the characteristics of human faces, while the extrinsic style code characterizes the structure and color styles from the external artistic portrait image. The structure style degree $d_s$ and the color style degree $d_c$ determines the intensity of the style applied, *i.e.*, 0 for no stylization and 1 for full stylization. For clarity, we will use the style code to refer to the extrinsic style code in our paper.

*4.2.1 Style-Controllable Network Architecture.* The exemplar-based framework mainly follows the collection-based framework in Sec. 4.1, with two modifications to realize the flexible style control as in DualStyleGAN:

*Structure Style Control with Modified ModRes.* As in the collection-based framework, the right part of $G$ in Fig. 9 aims to upsample the content feature and render colors and textures. Differently, the color style code $d_c$ could come either from the input frame or from an external style image for color transfer. In DualStyleGAN, Modulative ResBlock (ModRes) is proposed to adjust the structure styles of the content features based on the style code from the external style image. To realize such structure transfer in VToonify, we introduce ModRes into our framework (the red blocks in Fig. 9) to adjust the ResBlock features of $E$. However, the original ModRes is applied to the low-resolution layers of 8×8, 16×16 and 32×32[1]. Its reception field and feature scale do not fully match the ResBlock features of $E$ (which are of 32×32 during training). To solve this issue, we modify the convolutional layers in each ModRes to their dilated version to re-scale the convolution kernels. Specifically, the dilation rates are 4, 2 and 1 for the layers corresponding to 8×8, 16×16 and 32×32 in DualStyeGAN, respectively.

One advantage of this modification is that we do not need to alter the pre-trained parameters of ModRes, which can be directly loaded into VToonify. Moreover, empowered by ModRes, our framework is able to support hundreds of fine-level styles in a style collection within a single model, providing users with more options.

*Style-Degree-Aware Fusion Modules.* The structure style degree $d_s$ determines whether the stylized face matches the style better or retains more original face features as in the top right of Fig. 1. The ability to flexibly adjust to which degree the original facial features are preserved is highly desired in real applications since every user has their own preferences. Our generator built upon DualStyleGAN inherits this function, making it possible to adjust the structure degree within a single model. Specifically, the ModRes produces residual features for structure adjustment, which are multiplied by $d_s$ and added to the ResBlock features of $E$ to generate style-degree-aware content features.

Besides the style-degree-aware content features, the mid-layer features of $E$ to characterize the original facial features are also

[1]We empirically find that ModRes in 4×4 layers of DualStyleGAN to be of little use due to the extremely low resolution. So we omit it in VToonify.

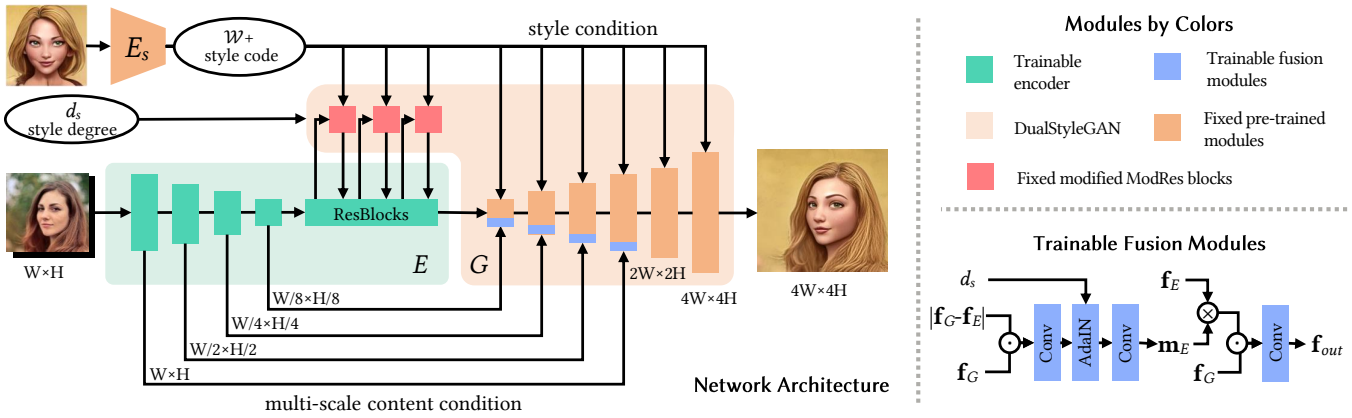



Fig. 9. **Framework of the proposed exemplar-based portrait video style transfer.** We use the highest 11 layers of the pre-trained DualStyleGAN and the modified ResMod blocks as our generator $G$. In addition to the extrinsic style code and the style degree $d_s$ as conditions, $G$ is also conditioned on the multi-scale content features from the encoder $E$. In the training phase, we use 256×256 images as input, while during testing, our fully convolutional network accepts video frames of various sizes $W \times H$. Input video: ©Pexels Andrea Piacquadio.

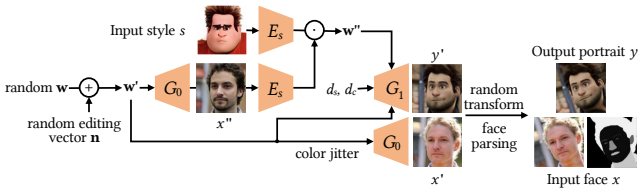
passed to $G$. Intuitively, these features should be extensively used for a low $d_s$ and be selectively used for a high $d_s$. To this end, we propose a style-degree-aware fusion module as illustrated in the bottom right of Fig. 9. Compared to the collection-based model, our exemplar-based model predicts an extra mask $\mathbf{m}_E$ to select the valid regions of $\mathbf{f}_E$ for fusion. Specifically, $\mathbf{f}_G$ and $\mathbf{f}_E$'s mismatch with $\mathbf{f}_G$ are concatenated channel-wisely and go through a convolutional layer. The result is then modulated using AdaIN conditioned by $d_s$ and fed into another convolutional layer to predict the one-channel attention mask $\mathbf{m}_E$. $\mathbf{f}_E$ is element-wisely multiplied by $\mathbf{m}_E$ and finally fused with $\mathbf{f}_G$.

#### 4.2.2 *Data Generation and Task-Oriented Training Objectives.*

*Data Generation.* We generate paired data based on DualStyleGAN to train our exemplar-based model. The data process is intuitively illustrated in Fig. 10. Here, $G_0$ and $G_1$ represent StyleGAN and DualStyleGAN to generate the real human face $x$ and artistic portrait $y$ in the style of the reference image $s$, respectively. As in Sec. 4.1.2, after sampling and applying attribute editing, we obtain the style code $\mathbf{w}'$ and its corresponding face image $x'' = G_0(\mathbf{w}')$. We further combine the color and texture style part of its embedded style code $E_s(x''_{\downarrow})$ with the structure style part of $E_s(s)$ from the style image $s$, resulting the extrinsic style code $\mathbf{w}''$ for DualStyleGAN. Together with the intrinsic style code $\mathbf{w}'$ and the style degrees $d_s, d_c$, the stylized portrait is obtained by $y' = G_1(\mathbf{w}', \mathbf{w}'', d_s, d_c)$. To learn color transfer, we apply color jittering to $\mathbf{w}'$ by style swapping [Karras et al. 2019] (replacing the last 11 vectors of $\mathbf{w}'$ with those from another style code) and generate a face image $x'$ that is inconsistent with $y'$ in color. Finally, after geometric transformations and face parsing, we obtain the paired data $\{x, y\}$ and their style code $\mathbf{w}''$ and style degree $d_s$.

*Model Initialization.* As in Sec. 4.1.2, we initialize the fusion modules to map $\mathbf{f}_G \odot (\mathbf{f}_E \otimes \mathbf{m}_E)$ to $\mathbf{f}_G$ and pre-train $E$ to minimize

$$\mathcal{L}_{\text{E}} = \|\mathbf{f}_E^{(\text{last})}(x''_{\downarrow}, \mathbf{w}'', d_s) - \mathbf{f}_{G_1}^{(8)}(\mathbf{w}', \mathbf{w}'', d_s)\|_2, \tag{8}$$




(b) Paired data generation for training exemplar-based portrait style transfer

Fig. 10. **Pipeline of paired training data generation** for training exemplar-based portrait style transfer.

where $E$ is additionally conditioned on $\mathbf{w}''$ and $d_s$ via ModRes. Note that the pre-training only involves the structure transfer, so the color style degree $d_c$ is omitted in Eq. (8).

*Task-Oriented Training Objectives.* Our exemplar-based model is also driven by the reconstruction loss, the adversarial loss and the temporal consistency loss introduced in Sec. 4.1.2. We use the class-conditional version [Karras et al. 2020a] of the discriminator to accept the conditions $d_s$ and $\mathbf{w}''$. According to different application requirements, specific training settings could be applied:

- **Control structure styles.** In applications where users would like to navigate around various structure styles within a single model, we just sample different style images $s$ from the style collection to generate training data. Otherwise, using a fixed $s$ will train a model specialized in that style.
- **Adjust structure style degree.** In applications where users would like to navigate around results under different structure style degrees within a single model, we just sample $d_s \in [0, 1]$ when generating data. We further regularize the sparsity of the attention mask $\mathbf{m}_E$ with a new loss term:

$$\mathcal{L}_{\text{mask}} = \lambda_{\text{mask}} \max\{\|\mathbf{m}_E\|_1 / |\mathbf{m}_E| - g(d_s), 0\}, \tag{9}$$

where $|\mathbf{m}_E|$ is the element number of $\mathbf{m}_E$ and $g(\cdot)$ is a monotonically decreasing function for $d_s \in [0, 1]$. Intuitively, a

large $d_s$ leads to a small $g(d_s)$, making the model predicts sparser $\mathbf{m}_E$, *i.e.*, using less information from the input frame to give room for greater facial structure adjustment.

- **Control color and texture styles.** In applications where users would like to mimic the color of the reference style image, we just set $d_c = 1$ when generating training data. This setting inherently supports the control of color style degree. As what $d_c$ does in DualStyleGAN, we just need to interpolate the color style codes from the style image and from the input frame during testing. Otherwise, we set $d_c = 0$ and turn off color jittering (*i.e.*, $x'$=$x''$), making the model well preserve the color and texture of the frames.

Note that the above training settings can be flexibly combined.

Obviously, training a model to support versatile stylization will inevitably sacrifice its quality. However, we can still provide a user friendly interaction with good trade-off between the quality and flexibility. During user interaction, the most versatile model can first provide visual previews without the need to load different checkpoints or retrain the network. After finding the ideal setting, users can then load the model specialized in that setting to obtain the high-quality, stylized output. Considering that training on a new setting requires about one hour for VToonify, retraining a new mode is also acceptable. Furthermore, models under commonly used settings can be pre-saved for reuse.

## 5 EXPERIMENTAL RESULTS

### 5.1 Implementation Details

*Training.* The experiment is conducted on NVIDIA Tesla V100 GPUs. We use eight GPUs to train VToonify, where pre-training encoder and training full VToonify use a batch size of 1 and 4 per GPU, respectively. We pre-train the encoder for 30,000 iterations, taking about 50 minutes. We train VToonify for 2,000 iterations, taking about 55 minutes. Note that for each style collection, the encoder only needs to be pre-trained once. It can be shared among different training settings in Sec. 4.2.2. The paired data is generated along with the training. Thus the above reported training time includes the time for data generation. The weights to balance different loss terms are set as $\lambda_{\text{mse}} = 0.1$, $\lambda_{\text{perc}} = 0.01$, $\lambda_{\text{adv}} = 0.01$, $\lambda_{\text{tmp}} = 1$, and $\lambda_{\text{mask}} = 0.0005$. $g(d_s) = 0.1 + 0.9(1 - d_s)^2$. To smooth the face parsing map, we use $k = 5$, and set $\sigma_t = 5.5$ and $\sigma_s = 0.2$. We use a pre-trained pSp [Richardson et al. 2021] as our style encoder $E_s$.

Since our method has several training settings for different tasks, we name each model individually, as summarized in Table 1 for clarify. For example, VToonify-D without suffix means a model built upon DualStyleGAN specialized for one exemplar style, while VToonify-Dsd indicates a versatile model supporting both structure style control and style degree adjustment.

*Testing.* For each video, we follow [Karras et al. 2019] to detect, align the face in the first frame and crop 256×256 images for pSp to obtain the style code. This style code is used for all other frames. We apply BiSeNet [Yu et al. 2018] to extract face parsing map for each frame. We find the original parsing maps work well for most of the videos. Thus by default, we do not apply the face parsing map smoothing algorithm in the experiment. It is optional for videos with inaccurately predicted parsing maps. We use the FaceForensics++ [Rössler et al. 2019] and videos from Pexels as our testing dataset. We proprocess the original videos by detecting the face in the first frame, and resizing the frame so that the eye distance is 64 pixels. Centered on the eyes, we crop the first frame to almost 400×400. All other frames use the same resizing and cropping parameters as the first frame to generate the final video for testing.

### 5.2 Flexible Style Control

This section elaborates the flexibility of VToonify in style control. We compare the performance of our model trained on different tasks on the same video clip, as shown in Fig. 11. In Fig. 11(b), VToonify-D specific to $d_s = 0.5$ and a fixed style synthesizes high-quality frames. Figures 11(c)-(f) show the results of VToonify-Ds, VToonify-Dd, VToonify-Dsd and VToonify-Dsdc, respectively. The conditional inputs of $d_s$ and the reference style image are shown in the lower left of frame#0 and frame#180. All models use the same condition as VToonify-D in frame#90 for quality comparison. It can be seen that VToonify-Ds supports fine-level control by choosing exemplar style images, and the facial structure style gradually becomes more adorable in Fig. 11(c). VToonify-Dd supports adjusting the style degree. With $d_s = 0$, our model purely super-resolves the input frame, while $d_s = 1$ leads to strong styles. Here, setting $d_s = 0.5$ provides a balance in facial feature preservation and artistry. VToonify-Dsd is a versatile model to support both of the above controls. From the local regions, it can be seen that although the capacity of these models differs, they demonstrate similar performance, possibly due to the good distillation of the backbones. The main difference is that the specialized VToonify-D can better preserves the details.

The main sacrifice of the quality comes from color style control, as in Fig. 11(f). In this case, since the training pairs have distinct color and texture discrepancies, the model has to rely more on the style condition than the content condition. As a result, VToonify-Dsdc cannot preserve the details, such as the double eyelid. However, the facial region is general still of high quality as other models.

### 5.3 Comparison with State-of-the-Art Methods

*Comparison Methods.* Since there are few methods that exactly handle our task, we compare with the most related ones and adapt

Table 1. **Summary of VToonify models and settings**.

| Name (Backbone) | Settings |
|---|---|
| VToonify-T (Toonify) | collection-based model |
| VToonify-D∗ (DualStyleGAN) | exemplar-based model, with suffix ∗ of:<br>'s': one model for various structure styles<br>'d': one model for various style degrees<br>'c': one model for various color styles |

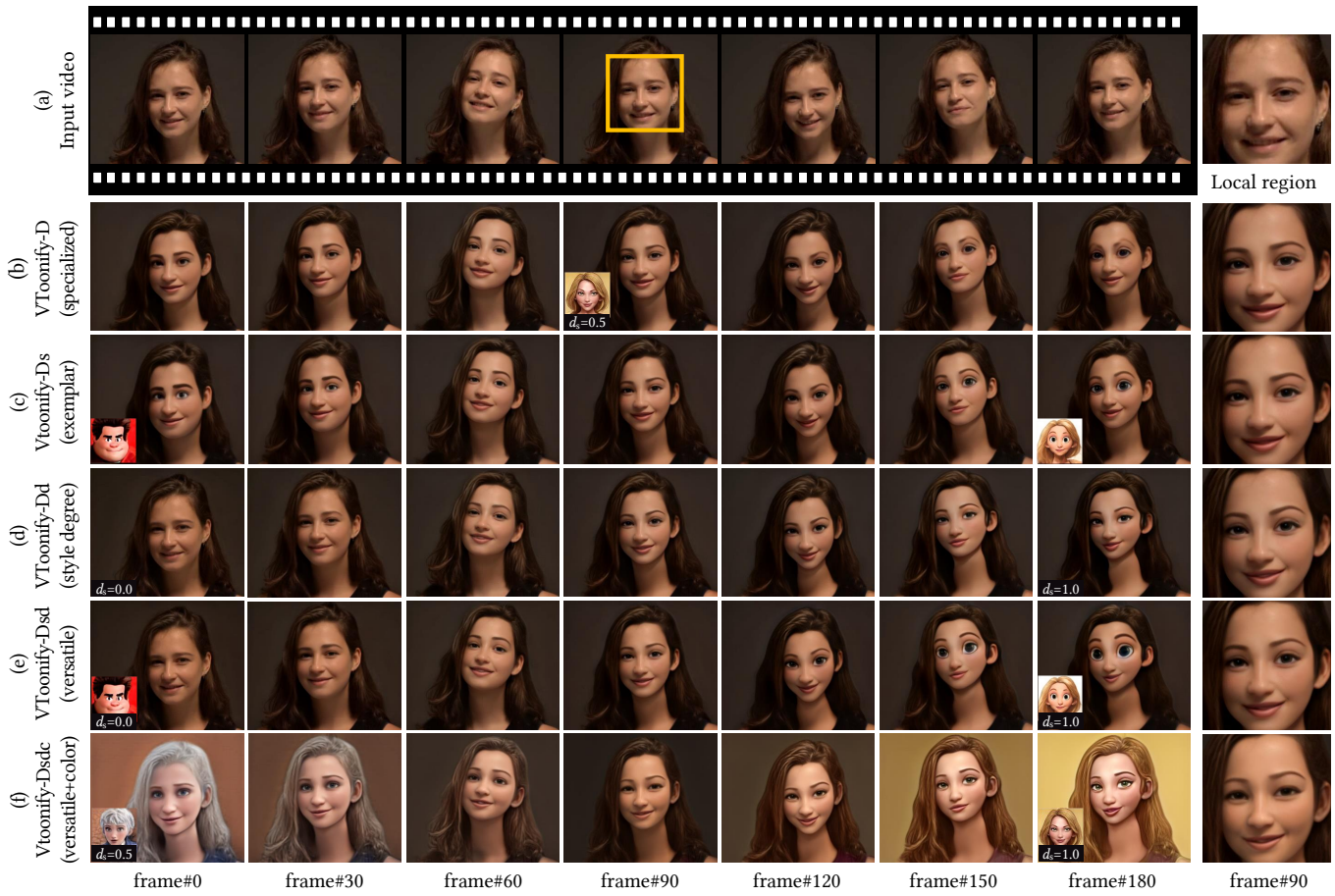



Fig. 11. Our method supports flexible style control on fine-level exemplar structure styles, color styles and style degree. Input video: ©Pexels Ketut Subiyanto.

them to our task. Besides our two backbones, we choose a high-resolution image-to-image translation baseline for StyleGAN distillation, and two image animation baselines for exemplar-based video generation. Specifically,

- **Toonify** [Pinkney and Adler 2020] is a StyleGAN-based backbone for collection-based style transfer on aligned faces. We follow [Karras et al. 2019] to align faces and crop 256×256 images for pSp to obtain the style codes. The style codes are fed into Toonify to obtain 1024×1024 stylized result. Finally we re-align the result back to its original position in the video. The unstylized region is simply set to black.
- **DualStyleGAN** [Yang et al. 2022] is a StyleGAN-based backbone for exemplar-based style transfer. We apply the same data preprocessing and postprocessing as in Toonify.
- **Pix2pixHD** [Wang et al. 2018] is an image-to-image translation model, which is widely used to distill pre-trained models for high resolution editing [Viazovetskyi et al. 2020]. We train it on our paired data. We follow pix2pixHD to use our extracted parsing map as its additional instance map inputs.
- **First Order Motion (FOM)** [Siarohin et al. 2019] is a representative image animation model. It is trained on 256×256 images. and has degraded performance using other image sizes. Therefore, we first resize the video frames to 256×256 for FOM to animate and resize the results to their original size. For a fair comparison, FOM uses the first stylized frame of our method as its reference style image.
- **DaGAN** [Hong et al. 2022] is an image animation model considering 3D face information. We apply the same data preprocessing and postprocessing as in FOM.

*Comparison with Face Style Transfer Methods.* Figure 12 shows the qualitative comparison with three high-resolution models: Toonify, DualStyleGAN and pix2pixHD. Since pix2pixHD is trained on paired data generated by DualStyleGAN in a specific style setting, we correspondingly use the specialized VToonify-D for comparison. It can be seen that VToonify-T and VToonify-D surpass their corresponding backbones Toonify and DualStyleGAN in stylizing the complete video, while maintaining the same high quality and visual features as the backbones for each single frame. For example, VToonify-T follows Toonify to impose a strong style effect like round eyes in the Pixar style. VToonify-D, on the other hand, is better at preserving the facial features. Compared with VToonify-D, pix2pixHD suffers


Fig. 12. **Visual comparison with face style transfer methods on four style collections**. Input video: ©Pexels Tima Miroshnichenko, Kampus Production, Anthony Shkraba, Rodnae Productions.

Table 2. **User preference rates** in terms of style similarity, content preservation, temporal consistency and overall quality.

| Method | Style | Content | Temporal | Overall |
|---|---|---|---|---|
| Toonify | 7.4% | 5.5% | 4.5% | 3.9% |
| Vtooniy-T | 31.3% | 21.6% | 29.4% | 31.9% |
| DualStyleGAN | 1.0% | 1.0% | 0.0% | 0.3% |
| pix2pixHD | 11.9% | 9.0% | 5.5% | 9.0% |
| VToonify-D | **48.4%** | **62.9%** | **60.6%** | **54.9%** |
| FOM | 8.4% | 5.8% | 6.8% | 4.5% |
| DaGAN | 1.6% | 3.6% | 2.2% | 1.6% |
| VToonify-Dsd | **90.0%** | **90.6%** | **91.0%** | **93.9%** |

from flickers and artifacts. For example, in the Caricature style, there are black holes near the cheek in pix2pixHD's results.

Since pix2pixHD and our method are not trained on real data, it is not feasible to use the common metrics like IS and FID. Therefore, we report user preference scores for quantitative evaluations. We conduct a user study, where 31 subjects are invited to select what they consider to be the best results from the five style transfer methods in terms of 1) how well the result matches the style of the style collection, 2) how well the result preserves the details of the original video, 3) the temporal consistency of the result and 4) the overall quality of video toonification. Ten testing videos from FaceForensics++ are used. Table 2 summarizes the average preference rates, where VToonify-D receives the best rates in all four metrics.

*Comparison with Image Animation Methods.* We present a visual comparison with FOM and DaGAN in Fig. 13. Since image animation frameworks can handle arbitrary style per model, we correspondingly use the versatile VToonify-Dsd for comparison. The first video has one challenge: new content. The image animation frameworks inherently cannot create information beyond its reference style image, *e.g.*, hands appearing in frame#7. By comparison, VToonify-Dsd is robust to the new content.

The second video has large motions. Although image animation methods successfully animate the stylized frame#0 based on the motions in the video, when the face expression changes drastically, FOM and DaGAN fail to generate the mouth details, leading to a


Fig. 13. **Visual comparison with image animation methods**. Input video: ©Pexels Anthony Shkraba and Rodnae Productions.

blurry face. By comparison, the results of our method show rich facial details consistently throughout the video.

To quantitatively evaluate VToonify-Dsd, we perform a user study in the same setting as VToonify-T and VToonify-D, where 31 subjects are invited to evaluate the results on ten testing videos from FaceForensics++. Table 2 reports the average preference rates, where VToonify-Dsd receives the best rates in all four metrics.

*Running Time.* Table 3 reports the running time (excluding video reading/writing) of the compared methods on a NVIDIA Tesla V100 GPU with a batch size of 1. Pix2pixHD is faster than the three frameworks based on the big StyleGAN model. The running time of VToonify is similar to Toonify and DualStyleGAN on aligned 1024×1024 videos. For unaligned 1600×1280 videos, Toonify and DualStyleGAN need to detect and crop faces every frame, making them less efficient than VToonify. VToonify surpasses FOM and DaGAN for generating higher-resolution videos in comparable time. Since our method handles every frame independently, it can be further accelerated by parallel processing.

## 5.4 Ablation Study

*Loss Functions.* Figure 14 compares the results with and without the adversarial loss $\mathcal{L}_{\text{adv}}$ in Eq. (5). Without $\mathcal{L}_{\text{adv}}$, the results have

Table 3. **Running time** on different video size (second per frame).

| **Output Size** | 1024×1024 | 1600×1280 | 256×256 |
|---|---|---|---|
| Toonify | 0.117 | 0.389 | - |
| DualStyleGAN | 0.128 | 0.402 | - |
| pix2pixHD | 0.015 | 0.017 | - |
| FOM | - | - | 0.169 |
| DaGAN | - | - | 0.059 |
| VToonify | 0.106 | 0.198 | - |

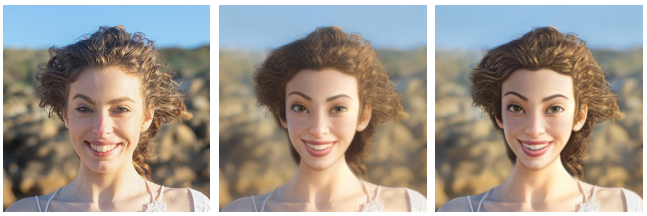



Fig. 14. **Effect of $\mathcal{L}_{\text{adv}}$** in reinforcing the realism of the output. Input video: ©Pexels Kampus Production.

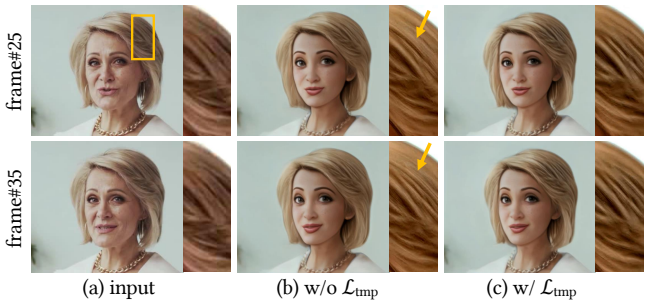



Fig. 15. **Effect of $\mathcal{L}_{\text{tmp}}$** in achieving temporal consistency. The local regions are enlarged with their contrast enhanced for better visual comparison. Input video: ©Pexels Anthony Shkraba.

low contrast and blurry hair details. The adversarial training effectively enriches the texture details, enhancing the overall realism. The effect of the temporal consistency term in Eq. (5) is shown in Fig. 15. Without $\mathcal{L}_{\text{tmp}}$, the same strand of hair is stylized into different textures in two consecutive frames, leading to annoying flickers. The proposed temporal consistency loss, though simple, solves this issue. In Fig. 16, we study the effect of the mask loss $\mathcal{L}_{\text{mask}}$ in Eq. (9). We compare the results with and without $\mathcal{L}_{\text{mask}}$, and show the estimated masks $\mathbf{m}_E$ in four fusion modules in $G$. The mask loss is proposed for style degree adjustment. For $d_s = 0$, both settings effectively leverage the mid-layer content features to reconstruct the input frame. However, when $d_s = 0.5$, the model without $\mathcal{L}_{\text{mask}}$ overuses the content features, leading to the ghosting artifacts at the edge of clothes. By comparison, the mask loss imposes the sparsity of $\mathbf{m}_E$, forcing the model to select only the valid information, *e.g.*,

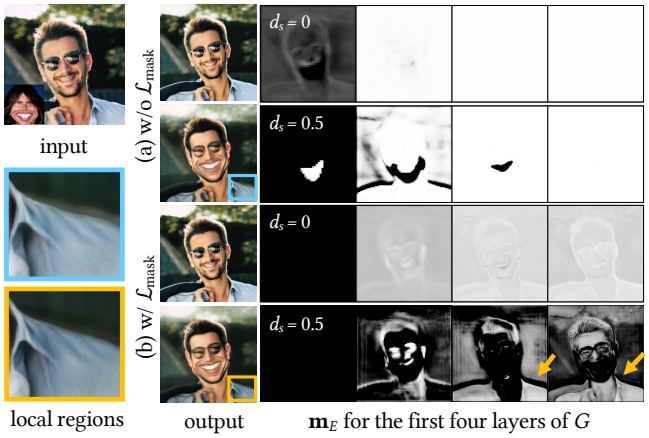



Fig. 16. **Effect of $\mathcal{L}_{\text{mask}}$** in eliminate ghosting artifacts. Input video: ©Pexels Chloe.

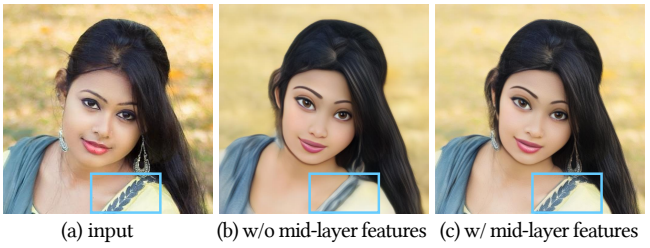



Fig. 17. **Effect of multi-scale content features** in preserving the visual details of the input frame. Input video: ©Pexels Joy Deb.

regions inside the clothes and forehead, avoiding the ghosting artifacts. Note that the edge of clothes has low values in $\mathbf{m}_E$ as pointed by the yellow arrows.

*Multi-Scale Content Features.* In addition to $\mathcal{L}_{\text{mask}}$, we further investigate the multi-scale content features in preserving the details. We train a baseline model by manually setting all $\mathbf{m}_E = \mathbf{0}$ to remove the mid-layer content features. As shown in Fig. 17(b), with only a single layer content feature, the hair and the patterns in the clothes becomes blurry. By using multi-scale content features, these important details, which are hard to encode and reconstruct within DualStyleGAN, are effectively synthesized.

*Encoder Pre-Training.* Figure 18 compares VToonify-Dsd trained with randomly initialized $E$ and with pre-trained $E$. Without pre-training, our model fails to capture the structure style of the reference style image. The pre-training strategy makes it easier to distill DualStyleGAN for effective exemplar-based style transfer.

*Parsing map.* Figure 19 compares VToonify-Dsd trained with and without face parsing maps as additional inputs. It can be seen that the face parsing map helps our model to generate more accurate structures of the face.

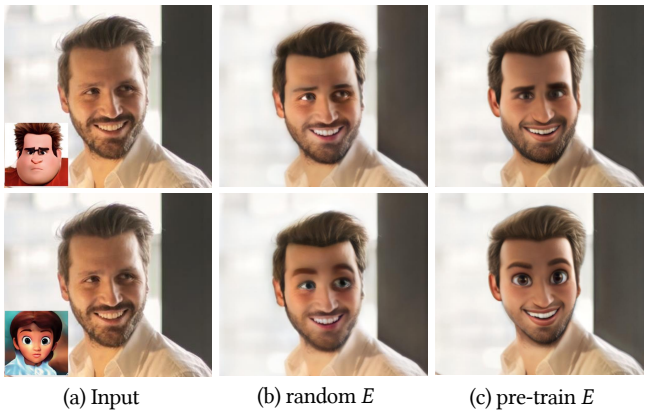



Fig. 18. **Effect of pre-training the encoder** in model distillation. Input video: ©Pexels Andrea Piacquadio.

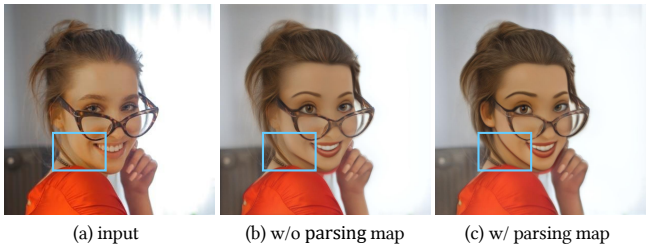
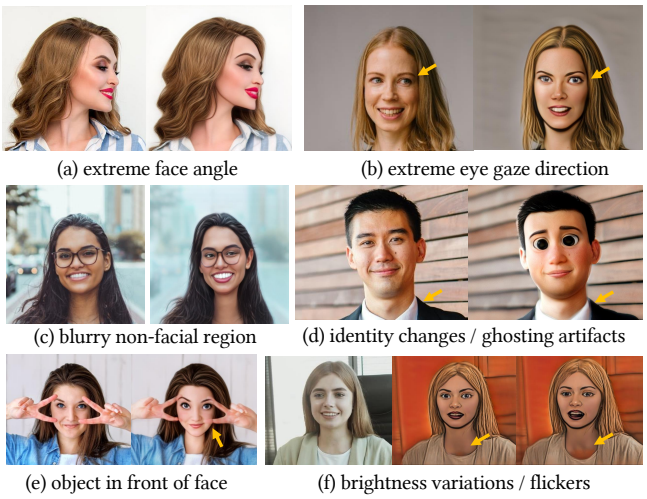



Fig. 19. **Effect of face parsing maps** in generating cartoon-like face structures. Input video: ©Pexels Andrea Piacquadio.

## 5.5 Limitations

In this section, we discuss limitations of our proposed method.

(1) **Data/model bias**. Our framework distills both the data and the model from the StyleGAN-based backbones, thus suffering the data bias and model bias. For example, as with most StyleGAN-based models, our method cannot handle faces in extreme angles (unable to adjust the facial structure in Fig. 20(a)) or with extreme eye gaze direction (wrong direction in Fig. 20(b)), either. Another example is that DualStyleGAN tends to synthesize images with blurry backgrounds under the Caricature style. Thus our method cannot preserve the details of non-facial regions in this style as shown in Fig. 20(c). The bias could also lead to failure of reference style preservation. For example, as reported in DualStyleGAN, uncommon styles like the extremely large cartoon eyes cannot be well imitated. The style is sometimes not respected with the original color styles. For example, the results in Fig. 11(b)-(e) look photo-realistic, less similar to the reference cartoon style. It is difficult to overcome these limitations inherited from our backbones. But in turn, our framework benefits from stronger backbones, which is promising with the fast advancement in computer graphics and vision.



(a) extreme face angle (b) extreme eye gaze direction

(c) blurry non-facial region (d) identity changes / ghosting artifacts

(e) object in front of face (f) brightness variations / flickers

Fig. 20. **Failure cases of VToonify**. Input video: ©Pexels Anastasiya Gepp, Mentatdgt, Daniel Xavier, Pixabay, Tima Miroshnichenko.

(2) **Identity Preservation**. Unlike existing models for caricature generation [Cao et al. 2018], which aims to achieve artistry and identity preservation at the same time, in our task, we aim to change facial features such as the round cartoon eyes (Fig. 20(d)) following our backbones. Therefore, the identity might be not well preserved with a large $d_s$.

(3) **Ghosting artifacts**. The artifacts mainly come from the scale mismatches between the stylized facial region and remaining regions. For example, Pixar portrait has a large head and a thin neck. Therefore, there is a gap between the shrinked neck and the collar, leading to ghosting artifacts as in Fig. 20(d).

(4) **Occlusions**. Our method is less effective in handling objects in the facial region, *e.g.*, hands in Fig. 20(e). Unlike the non-facial region, our method relies less on the mid-layer content features in the facial regions as visualized in Fig. 16(b). One possible solution is to manually overlap random occlusions on the training images. Another similar problem is the missing of the small earrings in Fig. 20(c).

(5) **Color control with artifacts**. As pointed out in Sec. 5.2, enabling color control forces the model to rely less on the mid-layer content features, which not only losses the details but also enlarges the small flickers in the original video. We observe ignorable variations in saturated regions sometime lead to noticeable brightness variations in Fig. 20(f). In future work, we would like to explore a better way of color control, *e.g.*, combining our versatile model VToonify-Dsd with state-of-the-art color transfer methods.

(6) **Flickers**. Figure 20(f) also shows the flickers in the background. Our flicker suppression loss is based on a simple uniform optical flow, thus might be less effective on scenes with saturated regions or complex motions.

## 6 CONCLUSION

In this paper, we propose a new VToonify framework for style controllable high-resolution video toonification. By distilling StyleGAN-based image toonification models in terms of both their synthetic data and network architectures, our framework presents high performance in handling videos and provides flexible control over the structural style, color style and style degree. While existing StyleGAN-based image editing methods focus on encoding faces into the style latent space for processing, we show that combining the high-level style code and multi-level content features with spatial resolution can better reconstruct the image details, especially for non-facial objects. It also overcomes StyleGAN's inherent limitation of fixed resolution and aligned faces. We believe the idea of our framework design is not limited to face style transfer, which can be potentially applied to other image and video editing tasks such as image super-resolution and facial attribute editing.

## ACKNOWLEDGMENTS

The authors would like to thank Yuming Jiang for proofreading. This study is supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

## REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. 2019. Image2stylegan: How to embed images into the stylegan latent space?. In *Proc. Int'l Conf. Computer Vision*. 4432–4441.

Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. 2021. Restyle: A residual-based stylegan encoder via iterative refinement. In *Proc. Int'l Conf. Computer Vision*. 6711–6720.

Yuval Alaluf, Or Patashnik, Zongze Wu, Asif Zamir, Eli Shechtman, Dani Lischinski, and Daniel Cohen-Or. 2022a. Third Time's the Charm? Image and Video Editing with StyleGAN3. arXiv:2201.13433 [cs.CV]

Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. 2022b. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 18511–18521.

Kaidi Cao, Jing Liao, and Lu Yuan. 2018. CariGANs: unpaired photo-to-caricature translation. *ACM Transactions on Graphics* 37, 6 (2018), 1–14.

Kelvin CK Chan, Xintao Wang, Xiangyu Xu, Jinwei Gu, and Chen Change Loy. 2021. GLEAN: Generative Latent Bank for Large-Factor Image Super-Resolution. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*.

Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. 2017. Coherent online video style transfer. In *Proc. Int'l Conf. Computer Vision*. 1105–1114.

Jie Chen, Gang Liu, and Xin Chen. 2019. AnimeGAN: A novel lightweight gan for photo animation. In *International Symposium on Intelligence Computation and Applications*. Springer, 242–256.

Min Jin Chong and David Forsyth. 2021. GANs N' Roses: Stable, Controllable, Diverse Image to Image Translation. *arXiv preprint arXiv:2106.06561* (2021).

Tan M Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. 2022. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 11389–11398.

Gereon Fox, Ayush Tewari, Mohamed Elgharib, and Christian Theobalt. 2021. Stylevideogan: A temporal generative model using a pretrained stylegan. In *Proc. British Machine Vision Conference*.

Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022. StyleGAN-NADA: CLIP-guided domain adaptation of image generators. *ACM Transactions on Graphics* 41, 4 (2022), 1–13.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 2414–2423.

Agrim Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2017. Characterizing and improving stability in neural style transfer. In *Proc. Int'l Conf. Computer Vision*. 4067–4076.

Erik Härkönen, Aaron Hertzman, Jaakko Lehtinen, and Sylvain Paris. 2020. GANSpace: Discovering Interpretable GAN controls. In *Advances in Neural Information Processing Systems*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 770–778.

Fa-Ting Hong, Longhao Zhang, Li Shen, and Dan Xu. 2022. Depth-Aware Generative Adversarial Network for Talking Head Video Generation. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*.

Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. 2017. Real-time neural style transfer for videos. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 783–791.

Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 5967–5976.

Wonjong Jang, Gwangjin Ju, Yucheol Jung, Jiaolong Yang, Xin Tong, and Seungyong Lee. 2021. StyleCariGAN: caricature generation via StyleGAN feature map modulation. *ACM Transactions on Graphics* 40, 4 (2021), 1–16.

Yuming Jiang, Ziqi Huang, Xingang Pan, Chen Change Loy, and Ziwei Liu. 2021. Talk-to-edit: Fine-grained facial editing via dialog. In *Proc. Int'l Conf. Computer Vision*. 13799–13808.

Justin Johnson, Alexandre Alahi, and Fei Fei Li. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. European Conf. Computer Vision*. Springer, 694–711.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2020a. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems*, Vol. 33. 12104–12114.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems* 34 (2021).

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 4401–4410.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020b. Analyzing and improving the image quality of stylegan. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 8110–8119.

Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwang Hee Lee. 2019. U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation. In *Proc. Int'l Conf. Learning Representations*.

Bing Li, Yuanlue Zhu, Yitong Wang, Chia-Wen Lin, Bernard Ghanem, and Linlin Shen. 2021. AniGAN: Style-Guided Generative Adversarial Networks for Unsupervised Anime Face Generation. *IEEE Transactions on Multimedia* (2021).

Feng-Lin Liu, Shu-Yu Chen, Yu-Kun Lai, Chunpeng Li, Yue-Ren Jiang, Hongbo Fu, and Lin Gao. 2022. DeepFaceVideoEditing: Sketch-Based Deep Editing of Face Videos. *ACM Transactions on Graphics* 40, 4 (2022), 1–16.

Ori Nizan and Ayellet Tal. 2020. Breaking the cycle-colleagues are all you need. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 7860–7869.

Gaurav Parmar, Yijun Li, Jingwan Lu, Richard Zhang, Jun-Yan Zhu, and Krishna Kumar Singh. 2022. Spatially-Adaptive Multilayer Selection for GAN Inversion and Editing. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 11399–11409.

Justin NM Pinkney and Doron Adler. 2020. Resolution Dependent GAN Interpolation for Controllable Image Synthesis Between Domains. *arXiv preprint arXiv:2010.05334* (2020).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proc. IEEE Int'l Conf. Machine Learning*. PMLR, 8748–8763.

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 *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*.

Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. 2022. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics* (2022).

Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2019. FaceForensics++: Learning to Detect Manipulated Facial Images. In *Proc. Int'l Conf. Computer Vision*.

Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2016. Artistic style transfer for videos. In *German conference on pattern recognition*. Springer, 26–36.

Ahmed Selim, Mohamed Elgharib, and Linda Doyle. 2016. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics* 35, 4 (2016), 1–18.

Xuning Shao and Weidong Zhang. 2021. SPatchGAN: A Statistical Feature Based Discriminator for Unsupervised Image-to-Image Translation. In *Proc. Int'l Conf. Computer Vision*. 6546–6555.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. 2020. Interpreting the latent space of gans for semantic face editing. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 9243–9252.

Yujun Shen and Bolei Zhou. 2021. Closed-form factorization of latent semantics in gans. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 1532–1540.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. 2019. First order motion model for image animation. *Advances in Neural Information Processing Systems* 32.

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* 40, 4 (2021), 1–13.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. European Conf. Computer Vision*. Springer, 402–419.

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

Rotem Tzaban, Ron Mokady, Rinon Gal, Amit H. Bermano, and Daniel Cohen-Or. 2022. Stitch it in Time: GAN-Based Facial Editing of Real Videos. arXiv:2201.08361 [cs.CV]

Yuri Viazovetskyi, Vladimir Ivashkin, and Evgeny Kashin. 2020. Stylegan2 distillation for feed-forward image manipulation. In *Proc. European Conf. Computer Vision*. Springer, 170–186.

Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. 2022. High-fidelity gan inversion for image attribute editing. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. 11379–11388.

Ting Chun Wang, Ming Yu Liu, Jun Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*.

Wenjing Wang, Jizheng Xu, Li Zhang, Yue Wang, and Jiaying Liu. 2020. Consistent video style transfer via compound regularization, Vol. 34. 12233–12240.

Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. 2022. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

Shuai Yang, Liming Jiang, Ziwei Liu, and Chen Change Loy. 2022. Pastiche Master: Exemplar-Based High-Resolution Portrait Style Transfer. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*.

Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. 2021. A latent transformer for disentangled face editing in images and videos. In *Proc. Int'l Conf. Computer Vision*. 13789–13798.

Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. 2018. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proc. European Conf. Computer Vision*. Springer, 334–349.

Jiapeng Zhu, Ruili Feng, Yujun Shen, Deli Zhao, Zheng-Jun Zha, Jingren Zhou, and Qifeng Chen. 2021b. Low-rank subspaces in gans. In *Advances in Neural Information Processing Systems*, Vol. 34.

Jun Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proc. Int'l Conf. Computer Vision*. 2242–2251.

Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. 2021a. Barbershop: GAN-based image compositing using segmentation masks. *ACM Transactions on Graphics* 40, 6 (2021), 1–13.