Talking Head(?) Anime from a Single Image 4: Improved Model and Its Distillation

Pramook Khungurn
pixiv Inc.

November 30, 2023

Abstract

We study the problem of creating a character model that can be controlled in real time from a single image of an anime character. A solution to this problem would greatly reduce the cost of creating avatars, computer games, and other interactive applications.

Talking Head Anime 3 (THA3) is an open source project that attempts to directly address the problem [34]. It takes as input (1) an image of an anime character's upper body and (2) a 45-dimensional pose vector and outputs a new image of the same character taking the specified pose. The range of possible movements is expressive enough for personal avatars and certain types of game characters. However, the system is too slow to generate animations in real time on common PCs, and its image quality can be improved.

In this paper, we improve THA3 in two ways. First, we propose new architectures for constituent networks that rotate the character's head and body based on U-Nets with attention [23] that are widely used in modern generative models. The new architectures consistently yield better image quality than the THA3 baseline. Nevertheless, they also make the whole system much slower: it takes up to 150 milliseconds to generate a frame. Second, we propose a technique to distill the system into a small network (< 2 MB) that can generate 512×512 animation frames in real time (≥ 30 FPS) using consumer gaming GPUs while keeping the image quality close to that of the full system. This improvement makes the whole system practical for real-time applications.

1 Introduction

We are interested in animating a single image of an anime character through specifying explicit pose parameters, as if controlling a rigged 3D model. Our motivation is the recent popularity of *virtual YouTubers* (VTubers), which are anime characters that are performed in real time by actors with the help of recent computer graphics technologies [44]. Typically, VTubers employ controllable layered images (aka 2.5D models) [41] created by software such as Live2D [42], E-mote [45], and Spine [17]. Because such a model can be costly to create, a solution would make it much easier to acquire a controllable avatar and to produce computer games and other interactive media.

The problem has received some attention from the research community [90, 35], private companies [26, 1], and individual open-source developers [34, 80]. In particular, Khungurn proposes a neural network system called "Talking Head Anime 3" (THA3) that can generate simple animations of a humanoid anime character, given only a single image of the frontal view of the character's torso [34]. When the system is run on a powerful GPU, the character can be controlled interactively through 45 parameters, enabling rich facial expressions and rotation of the head and the body by small angles. With NO manual modeling, the system can generate movements that are similar to what typical hand-made VTuber models are capable of.

Nevertheless, the THA3 system has two main limitations. The first is the quality of the generated images. When occluded parts of the character are rotated and become visible, they are often blurry. Moreover, the system has a tendency to remove thin structures, such as hair stands after head rotation. The second is

the system's speed: interactive frame rates can only be achieved when using very powerful GPUs, such as the Nvidia Titan RTX [34]. As a result, it is not yet a practical system for real-time applications such as computer games or VTuber streaming.

This paper proposes two improvements to the THA3 system to address the two shortcomings above. The first improvement is a new architecture for the subnetworks that rotate body parts. THA3 uses an encoder-decoder network and a vanilla U-Net as described in the original paper [65]. Our new architecture is based on the variant of U-Net with attention [81], commonly used in diffusion probabilistic models [23, 16]. It improves image quality under three commonly used metrics, reduces blurring in disoccluded[1] areas, and preserves thin structures better. Unfortunately, it sacrifices inference speed to do so, taking more than 100 ms to generate a frame on the Titan RTX.

The second improvement directly addresses the speed problem. The idea is to distill [22] the knowledge of the full system into a student neural network is small ($\approx$ 2 MB) and can generate a 512×512 frames in no more than 30 ms using a consumer gaming GPU. However, the student is specialized to a specific input image and cannot animate any other. For a given character image, distillation takes several tens of hours, but, once the process is finished, the student network can be used as a controllable character model. This capability makes the THA system usable in real-time applications for the first time. While we do lose the ability to change character and animate it immediately, the system remains practical because a VTuber or a game character does not change its appearance so often (every second or every minute).

The architecture for the student network is based on the SInusoidal REpresentation Network (SIREN) [72], which we extend to make it faster and better at preserving details of the input image. To make SIREN faster, we make it generate images in a multi-resolution fashion similar to the way a GAN generates an image [62]: the first few layers generate a low-resolution feature tensor, which is upscaled and passed to later layers. To better preserve the details of the input image, we have the SIREN generate appearance flow [95] that is used to warp the input image. The result is then combined through alpha blending with another less detailed output, also produced directly by SIREN. We demonstrate through ablation studies that our proposed architecture achieves a good trade-off between speed and accuracy, being able to reproduce high-frequency details while generating large images in real time. We also propose a three-phase training process that is effective at training our proposed model, and we verify that all of the phases are necessary to achieve better image quality.

2 Related Works

2.1 Implicit Neural Representation

The student network is a special case of **neural implicit representations** (INRs) where neural networks are used to approximate signals rather than functions that transform them. INRs often incorporate positional encoding [77] or have unconventional activation functions [72, 67] or network structures [71]. Researchers have applied INRs to signals such as images [75, 9], 3D surfaces [47, 57], and volume density coupled with radiance [52]. INRs can be used to build generative models of high-resolution 3D signals, which were hard to achieve previously [10, 6, 7, 69, 76].

While INRs can be used to directly represent articulated characters [14, 89, 59, 97], we take the view that our signal is a parameterized collection of images like Bemana et al. [2] rather than a deformable 3D shape. As a result, our INR employs the same image processing techniques such as warping and interpolation. However, our work is different from Bemana et al.'s in two ways. Firstly, the parameters of our image collection are blendshape weights and joint angles rather than those related to viewpoint, lighting, and time. Secondly, we use an INR to approximate the outputs of a bigger neural network rather than to fill in the gap between sparse measurements.

[1] "Disocclude" means "to cause to be no longer occluded." As far as we know, the word does not appear in standard dictionaries but has been used in a number of computer vision papers [56, 88].

2.2 Parameter-Based Posing of a Single Image

Overall, we want to create simple animations from a single image of a humanoid character. The input is an image of a subject (the **target** image), and we need to modify it so that the subject is posed according to some specification. Based on how pose is specified, the problem can be classified into **parameter-based posing** (explicitly by a **pose vector**), **motion transfer** (implicitly via an image or video of another subject), or **visual dubbing** (inferred from a spoken voice record). Our system solves parameter-based posing of a single image. As a result, we will review works that solve the same problem, excluding those that take videos or multiple images as input. To our knowledge, there are three approaches to the problem at hand.

2.2.1 Direct Modeling

We can create a controllable model of the subject's geometry from the target image. The common approach is to fit a **3D morphable model** (3DMM) [4, 5, 43, 39, 58, 55, 93] to the image. While earlier works are limited in controllability and only suitable for image manipulation [3, 5, 18], recent works provide much more control [19, 24, 40, 21, 13, 37]. A limitation of this approach is that parametric models often do not model all visible parts. For example, models specialized to the face might ignore the hair [40, 37, 37], the neck [24], or both [19].

While much research has been done on modeling from human photos, much less attention has been paid to other image domains. Saragih et al. construct a 3D model of a non-human face and then deform it to create animations [68], but they can only animate masks. Jin creates E-mode models from single anime-style images [28]. Chen et al. study 3D reconstruction from a single anime character's image [8] where the reconstruction can be later animated with the help of off-the-shelf components [87, 33].

2.2.2 Generative Modeling in the Latent Space

We first train a generative model that maps a **latent code** to an image, engineering it so that the output image is controllable through a pose vector. At test time, we fit a latent code to the target image.[2] Animation frames can then be generated by fixing the latent code and varying the pose vector.

Much research has been done on controlling the human face. Tewari et al. trains a network to alter latent codes of a StyleGAN [31, 30] according to 3DMM parameters [78] and later proposes a specialized algorithm to fit latent codes to portraits [79]. Using different methods, Kowalski et al. [36] and Deng et al. [15] train GANs, each of whose latent codes have parts that are explicitly controllable. Recent works extend EG3D [7], a 3D-aware GAN, so that the facial expression can be controlled [46, 86, 76]

2.2.3 Image Translation

Alternatively, we can view parameter-based posing as a special case of **image translation**: transforming an image into another according to some criteria. Isola et al. [27] present a general framework based on conditional generative adversarial networks (cGANs) [53], which is extended in various aspects by subsequent research [96, 11, 12]. Recently, researchers have also started exploring using diffusion models for the task [66, 85, 38].

Pumarola et al. create a network that modifies human facial features given an *Action Units* (AUs) encoding of a facial expression [61]. Ververas and Zafeiriou do the same but use blendshape weights instead of the AUs [82]. Ren et al.'s PIRenderer handles not only facial expression but also head rotation [63]. Zhang et al.'s SadTalker [92] can control a face image through 3DMM parameters by mapping them to facial landmark positions, which are then fed to Chen et al.'s face-vid2vid model [83] to move the input image. Nagano et al. design a conditional GAN that outputs a realistic facial texture, taking as input the target image and renderings of a template mesh whose expression can be freely controlled [54].

The THA3 system [34] that we build upon is based on Pumarola et al.'s facial feature manipulation technique and Zhou et al.'s image rotation technique [94]. Zhang et al. extended the first version of THA

[2] Optionally, the generative model can be fine-tuned to match the input image better [64].

[32] in order to support larger rotation angles [90]. Kim et al. created a dataset that can be used to train parameter-based posers such as PIRenderer so that they work on anime characters [35].

3 Baseline

THA3 as a whole is an image translator that takes as inputs (1) an 512×512 image of the "half body shot" of a humanoid anime character and (2) a 45-dimensional pose vector and then outputs a new image of the same character, now posed accordingly. The 45 parameters allow a character to not only express various emotions but also move its head and body like a typical professionally-created VTuber model. Out of the 45 parameters, 39 control the character's facial expressions, and 6 control rotation of the face and the torso.

The system is composed of 5 neural networks, and they can be divided into two modules. Three networks form a module called the **face morpher** whose duty is to alter the character's facial expression. We will not modify this module, but we will distill it into a smaller network in Section 5. The remaining two networks are called the **half-resolution rotator** and the **editor**. Together, they form a module called the **body rotator** whose duty is to rotate the head and the torso according to the 6 non-facial-expression parameters. The half-resolution rotator operates on a half-resolution (256×256) image obtained by downscaling the output of the face morpher. Its output is then upscaled to 512×512 and then passed to the editor to improve image quality before finally being returned to the user.

The two networks share the same overall structure. Each contains a backbone convolutional neural network (CNN): the half-resolution rotator uses an encoder-decoder network, and the editor uses a U-Net. Each backbone network outputs a feature tensor that has the same resolution as the input image. The feature tensor can then be used to perform three image processing operations:

1. **Warping.** The feature is transformed into an *appearance flow*, a map that tells for each pixel in the output which pixel in the input should data be copied from [95]. The appearance flow is then applied to the input image to get a warped version of it.
2. **Direct generation.** The feature tensor is directly transformed into pixel values. This operation is not limited by what is visible in the input image. It yields more plausible disoccluded parts but cannot preserve all the details in the visible parts.
3. **Blending.** The feature tensor is transformed into an alpha map, which can then be used to blend the results of other steps together to get the best features of both operations.

Outputs of the networks are generated using some combinations of the above operations.

4 Improved Network Architecture

One of the main problems of THA3 is image quality. When the body parts are rotated and disoccluded parts become visible, these parts can be blurry. Moreover, if such parts are thin, the system tend to remove them altogether. To alleviate the problem, we modify the networks in the body rotator module without significantly changing their functions.

4.1 New Body Rotator Architecture

There is no change to the interface of the half-resolution rotator. It still takes (1) the input image scaled to 256×256 and (2) a pose vector, and it still outputs an appearance flow and an image of the posed character, both at the 256×256 resolution. On the other hand, we slightly change the editor's interface. It now takes in both outputs of the half-resolution rotator, scaled up to 512×512, instead of taking just the appearance flow like THA3's editor. The overall structure of the modified body rotator is given in in Figure 1.

We changed all backbone networks to U-Nets with attention layers, which are now widely used in diffusion models [23, 16] and prove to be excellent at image generation. We also changed how the networks handle its inputs and outputs.

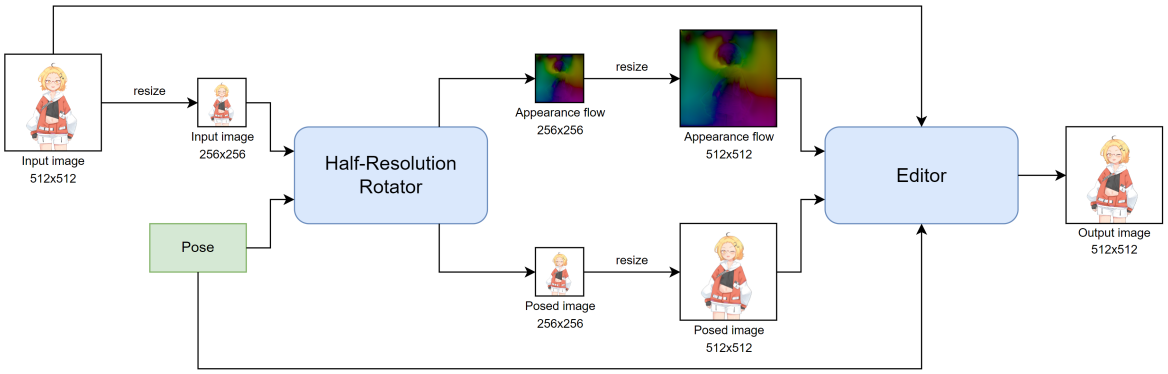

Figure 1: The modified body rotator module.

- The half-resolution rotator now generates a half-resolution posed image in three steps. It (1) warps the input image to generate one candidate output, (2) directly generates another candidate output, and (3) alpha blends the two results together. (See the "image formation" part of Figure 3.) In THA3, the alpha blending step is missing.
- The editor now has to take into account one additional input, so we make it fuse the input image and the two outputs of the half-resolution rotator with a convolution layer before processing the fused tensor with the backbone network. The rest of the network remains the same.

We refer the reader to the Appendix A for a more complete description of the changes.

4.2 Training

We use datasets created from approximately 8,000 controllable 3D anime character models we individually collected from the Internet. Each example in the datasets contains three items: (1) an image of a character in a "rest" post, (2) a pose vector, and (3) another image of the same character after being posed according to the pose vector. The training dataset contains 500,000 examples, while the test dataset contains 10,000. The two datasets do not share 3D models, ensuring clean separation between training and test data. Please refer to the write-up of the THA3 project for how to prepare the dataset [34].

We trained the networks using two types of losses: L_1 loss and the perceptual content loss [29]. The half-resolution rotator's training is divided into two phases where the first phase only uses the L_1 loss, and the second uses both. The editor's training has three phases. In the first two phases, it is trained like a rotator that operates on 512×512 images. In the last phase, we add to the network units that take into account the outputs of the half-resolution rotator and continue training. Technical details on the training process can be found in Appendix B. They include expressions for loss functions, training phases, weight initialization, optimizers, and learning rate schedules.

4.3 Results

4.3.1 Image Quality

We compare the new body rotator against the old THA3 body rotator. For qualitative comparison, we evaluate the networks by comparing the images they generate against the groundtruth images in the test dataset. We use three metrics for image similarity: (a) peak signal-to-noise ratio (PSNR), (b) structural similarity (SSIM) [84], and (c) LPIPS [91]. The averages of the metrics over the 10,000 examples of the test dataset are reported in Table 1. We can see that the use of U-Net with attention and the additional input

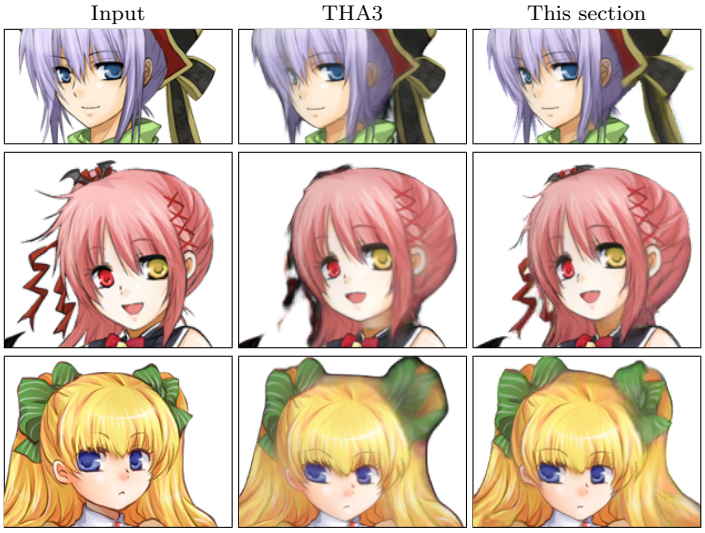

Figure 2: Qualitative comparison between images generated by body rotator models. The artworks were created by Mikatsuki Arpeggio [48, 51, 49, 50].

to the editor improve all the metrics. The LPIPS, in particular, sees an improvement of approximately 30% over THA3.

For qualitative comparison, we applied the networks to three hand-drawn characters, and we show the results in Figure 2. The characters' faces and bodies are rotated to the left of the viewer with the largest possible angles. For the 1st and 2nd characters, we can see that the THA3 rotator could not produce sharp left silhouettes of the faces, and the ribbons worn by the 2nd character are close to being completely erased. On the other hand, the architecture we propose generated much sharper silhouettes and preserved the ribbons better. For the 3rd character, the THA3 rotator generated blurry hair and ribbons on the right side, while ours generated sharper results. We can also see here that our proposed architecture preserved textures in the area better than the baseline.

Network	PSNR (↑)	SSIM (↑)	LPIPS (↓)
THA3 [34]	22.369330	0.909369	0.048016
Section 4	22.962184	0.919532	0.033566

Table 1: Quantitative comparison of body rotator models' performance.

4.3.2 Model Size and Speed

Table 2 compares the size and speed of THA3 system and our proposal. The new editor network is 4 times larger than the THA3 one, but it does not significantly increase the size of the whole system because there are four other networks that are already as large as it is. We assessed the system's speed by measuring the time it takes to fully process one input image and one pose, mirroring the situation in which it is used to generate one animation frame in an application. We performed experiments on three different computers, identified by the letters A to C, whose details are given in Appendix C. The computers have different GPUs, ranging from a research-oriented card to a consumer-oriented gaming one. From Table 2, we can see that our proposed architecture, while yielding higher image quality, were about 3 to 4 times slower than the THA3 system. In terms of number of frames per second (FPS), THA3 can in the best case[3] achieve around 30 FPS on a machine with very powerful GPUs, but the proposed architecture cannot even make 10 FPS under the same settings.

[3]FPS inside an application can be lower due to time spent on processing user inputs and updating UI.

System	Size (MB)			Time needed to generate a frame (ms)		
	HRR[4]	Editor	Total[5]	Computer A (RTX A6000)	Computer B (Titan RTX)	Computer C (GTX 1080 Ti)
THA3 [34]	128	33	517	35.899	41.409	64.607
This section	136	137	627	125.843	116.763	159.647

Table 2: Size and speed comparison between the THA3 system and our proposed one. The times needed to generate a frame are averages of 1,000 measurements.

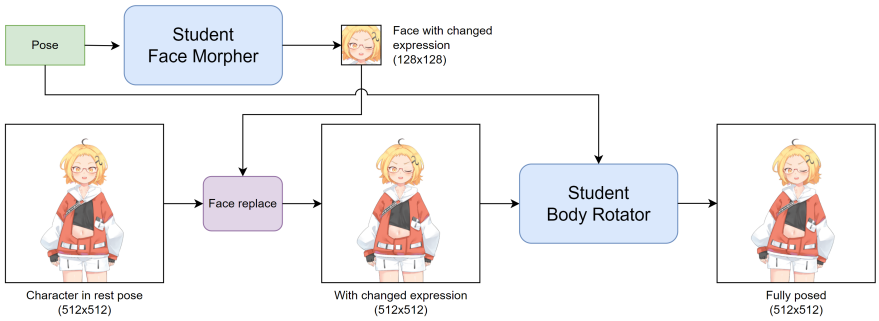

Figure 3: Overall architecture of the student model.

5 Distillation

Results from the last section reveal that our quest to improve image quality results in a much slower model. Our next task, then, is to improve image generation speed so that real-time performance is achieved on less powerful hardware.

We first observe that the system is overly capable. At any time, we can change the input image, and the change would be reflected on the output image immediately. Nevertheless, in computer games or in streaming, a character does not change its appearance every second or every minute. This functionality is thus unnecessary. By creating a model specialized to a particular input image, we may obtain a faster model that works under real-time constraints. If we prepare many such models in advance, we may swap the models to change characters or allow a character to change its appearance when needed.

To create such a specialized model, we rely on **knowledge distillation** [22], which is the practice of training a smaller model (the **student**) to mimic the behavior of a larger model (the **teacher**). In our case, the teacher is the full system as proposed in the last section.

All of the student models we proposed are coordinate-based networks [77] because, by construction, they allow generating any specific subimage at a cost proportional to the subimage's size. Moreover, unlike CNN-based image generators, subimage generation can be done without having to generate the whole image. This feature is beneficial for game characters and real-time streaming because, in some cases, the user might want to depict only the head instead of the whole torso. The specific architecture we employ is the SInusoidal REpresentation Network (SIREN) [72] because we found that it produced smooth images that fit well with the anime style. On the other hand, a competing approach (ReLU MLP with Fourier features [77]) tend to produce grainy artifacts [72].

5.1 Student Architecture

Like the full system, the student model contains two modules, **the face morpher** and the **body rotator**, with the same functionality. Instead of being a collection of five big networks, they are now two small

[4]HRR stands for "half-resolution rotator."
[5]Complete THA systems have three other networks. This column contains the sizes of all the networks combined.

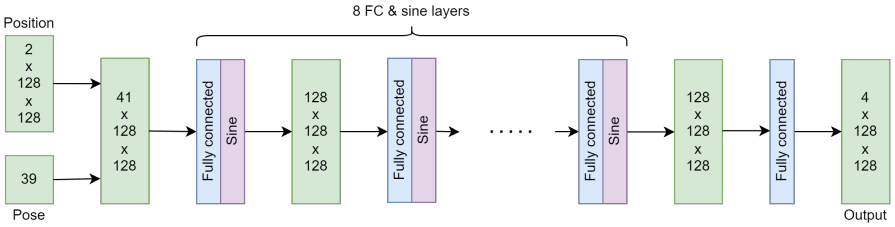

Figure 4: Architecture of the student face morpher.

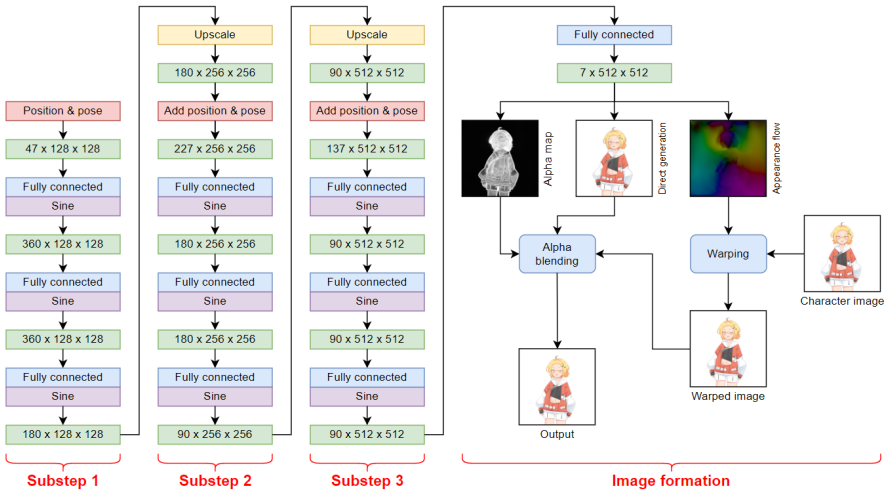

Figure 5: Architecture of the student body rotator.

networks whose total size is less than 2 MB. An overview of the student's architecture is shown in Figure 3.

The student face morpher is a SIREN with 9 fully connected layers where each hidden layer has 128 neurons. Its architecture is depicted in Figure 4. It is trained to generate a 128×128 area of the input image that contains the character's movable facial organs (eyebrows, eyes, mouth, and jaw). The SIREN receives as input a pixel position (2 dimensions) and a facial pose (39 dimensions), and it produces an RGBA pixel (4 dimensions). Its size is only 475 KB.

The student body rotator's architecture is more complicated because it needs to generate much larger outputs (512×512 images) in real time. A vanilla SIREN like the student face morpher would be too slow because it has to operate on tensors with a spatial size of 512×512 at all of its layers. To improve speed, we divide the image generation process into three substeps where the network would operate on tensors with spatial resolutions of 128×128 first, then 256×256, and lastly 512×512. Each substep has 3 fully connected layers, except for the last one which has 4, resulting in a network with 10 such layers. Moreover, in order to preserve fine details of the input image, the network does not generate the output image directly. Instead, it uses the image formation process employed by the teacher's body rotator. In particular, the network is trained to generate (1) an appearance flow, (2) an RGBA image, and (3) an alpha map. The output image is formed by first using the appearance flow to warp the input character image and then alpha blending the warped image with the directly generated RGBA image. The model size's is about 1.3 MB. The architecture of the student body rotator is depicted in Figure 5.

5.2 Student Training

5.2.1 Face Morpher

The student face morpher is trained to minimize the L1 differences between its outputs and those generated by the teacher face morpher. The loss function has two terms. The first is the L1 difference between the whole outputs, and the second is the L1 difference between areas that contain movable facial parts. We weigh the second term 20 times more than the first because the movable parts are small compared to the whole face. The precise definition of the loss is given in Appendix D.

At training time, the character image is fixed, and the pose vectors are sampled uniformly from the training dataset of the full system. Training lasts 2 epochs (1M examples), and the batch size is 8. We use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate starts at 10^{-4} and decays to 3.33×10^{-5}, 1×10^{-5} and then 3.33×10^{-6} after 200K, 500K, and 800K training examples, respectively. Training takes about an hour and a half on a computer with four V100 GPUs.

5.2.2 Body Rotator

Recall first that the body rotator uses the same image formation process as the teacher body rotator. The outputs of the last fully-connected layer are (1) an appearance flow I_{flow} which is immediately used to generate a warped image I_{warped} from the fixed character image, (2) an RGBA image I_{direct}, and (3) an alpha map I_{alpha}. Then, I_{warped}, I_{direct} and I_{alpha} are then combined with alpha blending to generate the final output image I_{final}. Because the teacher also generates these data as well, we distinguish between those generated by the student with the superscript "S" (such as I^S_{flow}, I^S_{warped}) and those generated by the teacher with the superscript "T" (such as I^T_{direct}, I^T_{final}).

The student body rotator is trained to minimize a 4-termed loss where each term involves one of the generated data above:

$$\mathcal{L}_{\text{br}} = \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{warped}}\mathcal{L}_{\text{warped}} + \lambda_{\text{direct}}\mathcal{L}_{\text{direct}} + \lambda_{\text{final}}\mathcal{L}_{\text{final}}$$

where $\mathcal{L}_{\square} = \|I^S_{\square} - I^T_{\square}\|_1$, and $\square$ can be replaced with the suffixes in the above equation. The λ-variables are weights that change throughout the training process, which is divided into three phases as shown in Table 3. We can see that the the first phase focuses on training the direct generation, the second on the warping, and the third on the final output.

Much like what we do with the student face morpher, we also sample pose vectors from the training dataset of the full system, use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and set the batch size to 8. However, training now lasts for 3 epochs (1.5M examples). Learning rate starts from 10^{-4} and decays to 3×10^{-5}, 10^{-5}, and 3×10^{-6} after we have shown 200K, 600K, and 1.3M training examples, respectively. Training takes about 10 hours on a computer with four V100 GPUs. We have not measured training time on other computers, but we surmise that it would take several ten hours on a machine with a single GPU.

Phase	# Examples	λ_{flow}	λ_{warped}	λ_{direct}	λ_{final}
#1	$\leq$ 400K	0.50	0.25	2.00	0.25
#2	$\leq$ 800K	5.00	2.50	1.00	1.00
#3	$\leq$ 1.5M	1.00	1.00	1.00	10.00

Table 3: Training phases of the student body rotator.

5.3 Results

We assess a model by using it to pose characters according to 1,000 fixed poses taken from the test dataset in Section 4.2. For each posed image, we compute the PSNR with respect to the corresponding image generated by the teacher model. We then record the average of the 1,000 resulting PSNR values.

5.3.1 Comparison Against the Teacher

System	Time needed to generate a frame (ms)		
	Computer A (RTX A6000)	Computer B (Tital RTX)	Computer C (GTX 1080 Ti)
THA3 [34]	35.899	41.409	64.607
Section 4	125.840	116.760	159.640
Student model	12.523	15.098	22.091

Table 4: Comparison between average time required to generate a frame of animation by the THA3 system, the teacher model (Section 4), and the student model.

Architecture	PSNR (dB)	Time needed to generate a frame (ms)		
		Computer A (RTX A6000)	Computer B (Tital RTX)	Computer C (GTX 1080 Ti)
Vanilla SIREN	38.259	21.319	33.086	54.937
Section 5.1 w/o multi-res	38.923	24.337	34.883	57.394
Section 5.1	38.881	12.523	15.098	22.091

Table 6: An ablation study on the architecture of the student model.

As mentioned in Section 5.1, the student model is much smaller than the full system: 1.8 MB versus 627 MB. It is also around 8 times faster to generate a single animated frame as can be seen in Table 4. Compared to the THA3 system, it is about 3 times faster and thus can now achieve real-time animation ($\geq$ 30 FPS) on Computer C, which has a standard consumer GPU.

Character	PNSR (dB)
Top [51]	36.156
Middle [49]	36.048
Bottom [50]	34.543

Table 5: Average PSNR of images generated by student models trained to animate the characters from Figure 2.

As for the quality of generated images, we trained student models on the three characters in Figure 2. We report the averaged PSNRs in Table 5, which range from 34 dB to around 36 dB. This means that the average error is about 2% of the maximum pixel value. Qualitatively, it is hard to spot large differences between outputs of the students and the teacher, but a student model might ignore extremely fine details such as the black dot that represents the nose as can be seen in Figure 6 and Figure 7.

5.3.2 Ablation Study on Student Network Architecture

In this section, we show that the proposed architecture yielded improvement over simpler alternatives. We compare our architecture against (a) a vanilla SIREN that generates the output image directly, and (b) our proposed architecture where the body rotator is modified so that it always operates at the 512×512 resolution. We trained the three architectures to animate a specific character image [74], and we evaluated them with the average PSNR metric. We also measured the average time it took to generate an animation frame on the 3 computers used in Section 4.3.2. The statistics are given in Table 6. For qualitative comparison, we show generated images in Figure 6.

We see in Table 6 that the student architectures' PSNR values are comparable to one another. However, Figure 6 reveals that the vanilla SIREN architecture is qualitatively much worse than the other two because it cannot reproduce fine face details, such as the eyebrows, the mouth shapes, and the highlights on the pupils. Preserving these fine details necessitates the more complicated image formation steps. The architecture without multi-resolution SIREN is slightly more accurate than the proposed architecture. However, it is very hard to spot differences between their generated images in Figure 6. The advantage of the proposed architecture is its speed: Table 6 shows that it is two times faster than the architecture without multi-resolution SIREN. In other words, the multi-resolution design maintains accuracy while making the network significantly faster.

Figure 6: Qualitative comparison between images generated by the teacher and three student architectures. The character is © Touhoku Zunko · Zundamon Project [73].

5.3.3 Ablation Study on Student Training Process

The training process for the student model has 3 training phases with different weights on loss terms. To show the necessity of the phases, we trained student models on the character image in the last section, ablating the training phases while keeping the rest of the settings the same. We report the models' average PSNR values in Table 7. We see that employing all phases yielded the best score. Omitting Phase #1 resulted in significantly worse image quality. This manifests qualitatively as noticeable differences in the shape of the rotated faces in Figure 7. Models that were trained with Phase #1 have PSNR scores of around 38, showing that they approximate the teacher's overall outputs well. However, there are visible degradations in the details. Model D did not reproduce the highlights on the pupils. Model E and Model F produced artifacts around the headband. Moreover, Model F also yielded jagged edges on one side of the head. Model G, which experienced all training phases, achieved the highest PSNR score and produced the least amount of artifacts, showing the necessity of all the training phases.

Model	Training phases			PSNR (dB)
	#1	#2	#3	
A			✓	29.308
B		✓		29.118
C		✓	✓	29.484
D	✓			38.026
E	✓		✓	38.668
F	✓	✓		37.399
G	✓	✓	✓	38.881

Table 7: Quantitative comparison between student models trained with and without specific training phases.

5.3.4 Miscellaneous Results

Student models are fast and lightweight that they can be executed inside a web browser and still generate animation in real time. In the supplementary material, we include two demo web applications. One allows the user to pose characters by manipulating UI widgets. The other makes characters imitate the user's movement as captured by a web camera.

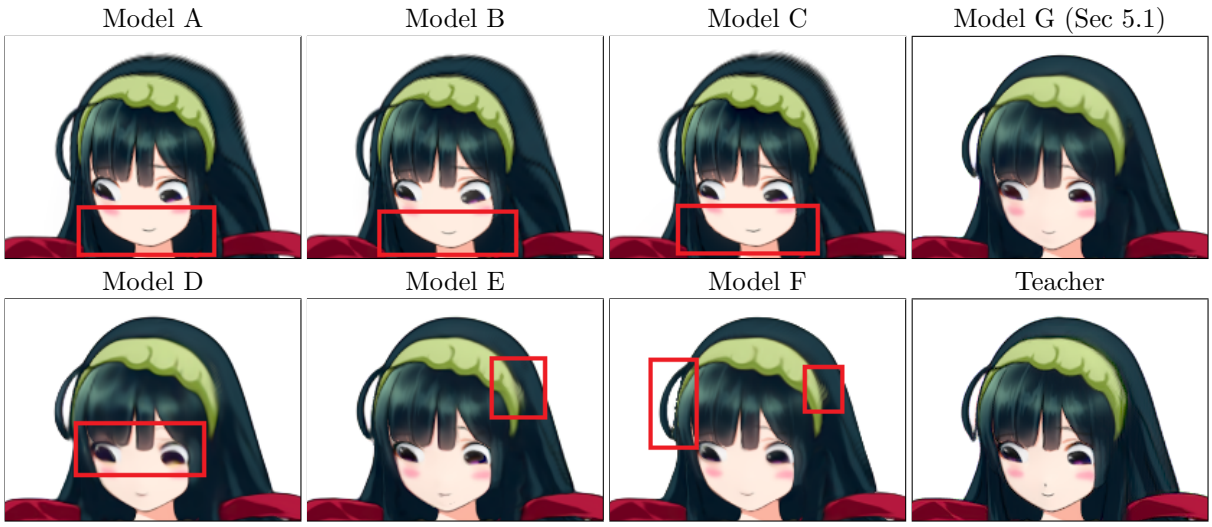

Figure 7: Qualitative comparison between outputs of models in Table 7. Problematic areas are highlight with red rectangles.

6 Conclusion

We proposed improvements to the Talking Head Anime 3 (THA3) system, increasing its image quality and speeding it up so that it can generate smooth animation in real time with a consumer gaming GPU. The latter improvement makes the system practical as a streaming tool for the first time. The main insight is that we can use a more expensive architecture (U-Net with attention) to get better image quality and then distill the improved model to small and fast students. Our technical contribution includes an effective architecture for the student model (multi-resolution SIREN with warping and blending) and an algorithm to train it.

There are still several limitations to our work. The image quality, while greatly improved by the new architecture for the body rotator, can still be improved further. The system can only move facial organs rotate of the face and the torso by small angles. Lastly, while the student model is lightweight enough to run on a consumer gaming GPU, it can still cannot be run on devices such as tablets or mobile phones. We hope to address these problems in future works.

References

[1] Algoage Inc. DeepAnime: Automatically turning illustration to anime. `https://lp.deepanime.com/`, 2022. Accessed: 2023-08-07.

[2] Bemana, M., Myszkowski, K., Seidel, H.-P., and Ritschel, T. X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020) 39*, 6 (2020).

[3] Blanz, V., Scherbaum, K., Vetter, T., and Seidel, H.-P. Exchanging Faces in Images. *Computer Graphics Forum* (2004).

[4] Blanz, V., and Vetter, T. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., p. 187–194.

[5] Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 413–425.

[6] Chan, E., Monteiro, M., Kellnhofer, P., Wu, J., and Wetzstein, G. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. In *CVPR* (2021).

[7] Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., Mello, S. D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T., and Wetzstein, G. Efficient geometry-aware 3D generative adversarial networks. In *CVPR* (2022).

[8] Chen, S., Zhang, K., Shi, Y., Wang, H., Zhu, Y., Song, G., An, S., Kristjansson, J., Yang, X., and Zwicker, M. Panic-3d: Stylized single-view 3d reconstruction from portraits of anime characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

[9] Chen, Y., Liu, S., and Wang, X. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 8628–8638.

[10] Chen, Z., and Zhang, H. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[11] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[12] Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020).

[13] Corona, E., Zanfir, M., Alldieck, T., Gabriel Bazavan, E., Zanfir, A., and Sminchisescu, C. Structured 3d features for reconstructing relightable and animatable avatars. In *CVPR* (2023).

[14] Deng, B., Lewis, J. P., Jeruzalski, T., Pons-Moll, G., Hinton, G., Norouzi, M., and Tagliasacchi, A. Nasa neural articulated shape approximation. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII* (Berlin, Heidelberg, 2020), Springer-Verlag, p. 612–628.

[15] Deng, Y., Yang, J., Chen, D., Wen, F., and Tong, X. Disentangled and controllable face image generation via 3d imitative-contrastive learning, 2020.

[16] Dhariwal, P., and Nichol, A. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems* (2021), M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., pp. 8780–8794.

[17] Esoteric Software. Spine: 2d animation for games. `http://esotericsoftware.com/`, 2023. Accessed: 2023-08-07.

[18] Fried, O., Shechtman, E., Goldman, D. B., and Finkelstein, A. Perspective-aware manipulation of portrait photos. *ACM Trans. Graph. 35*, 4 (July 2016).

[19] Geng, Z., Cao, C., and Tulyakov, S. 3d guided fine-grained face manipulation. In *CVPR* (2019).

[20] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[21] He, T., Xu, Y., Saito, S., Soatto, S., and Tung, T. Arch++: Animation-ready clothed human reconstruction revisited. In *ICCV* (2021).

[22] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

[23] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *CoRR abs/2006.11239* (2020).

[24] Hong, Y., Peng, B., Xiao, H., Liu, L., and Zhang, J. Headnerf: A real-time nerf-based parametric head model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[25] Huang, X., and Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV* (2017).

[26] IRIAM Inc. Character streaming service "iriam," using technology contributed by preferred networks, becomes the first in the world to implement automatic character modeling by ai in smart phones. an illustration can move with rich expression through the power of ai. `https://prtimes.jp/main/html/rd/p/000000006.000070082.html`, 2021. Accessed: 2023-08-07.

[27] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[28] Jin, Y. Crypko - a new workflow for anime character creation. `https://codh.repo.nii.ac.jp/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=400&item_no=1&page_id=30&block_id=41`, 2020.

[29] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision (ECCV)* (2016).

[30] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. Training generative adversarial networks with limited data. In *Proc. NeurIPS* (2020).

[31] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan, 2019.

[32] Khungurn, P. Talking head anime from a single image. `https://web.archive.org/web/20220401093041/https://pkhungurn.github.io/talking-head-anime/`, 2019. Accessed: 2023-08-04.

[33] Khungurn, P. Talking head anime from a single image 2: More expressive. `https://web.archive.org/web/20220327163627/https://pkhungurn.github.io/talking-head-anime-2/`, 2021. Accessed: 2023-08-04.

[34] Khungurn, P. Talking head(?) anime from a single image 3: Now the body too. `https://web.archive.org/web/20220606125417/https://pkhungurn.github.io/talking-head-anime-3/`, 2022. Accessed: 2023-08-04.

[35] Kim, K., Park, S., Lee, J., Chung, S., Lee, J., and Choo, J. Animeceleb: Large-scale animation celebheads dataset for head reenactment. In *Proc. of the European Conference on Computer Vision (ECCV)* (2022).

[36] Kowalski, M., Garbin, S. J., Estellers, V., Baltrušaitis, T., Johnson, M., and Shotton, J. Config: Controllable neural face image generation. In *European Conference on Computer Vision (ECCV)* (2020).

[37] Lattas, A., Moschoglou, S., Ploumpis, S., Gecer, B., Deng, J., and Zafeiriou, S. Fitme: Deep photorealistic 3d morphable model avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 8629–8640.

[38] Li, B., Xue, K., Liu, B., and Lai, Y.-K. Bbdm: Image-to-image translation with brownian bridge diffusion models. In *CVPR* (2023).

[39] Li, T., Bolkart, T., Black, M. J., Li, H., and Romero, J. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia) 36*, 6 (2017).

[40] Lin, C. Z., Nagano, K., Kautz, J., Chan, E. R., Iqbal, U., Guibas, L., Wetzstein, G., and Khamis, S. Single-shot implicit morphable faces with consistent texture parameterization. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023).

[41] Litwinowicz, P. C. Inkwell: A 2-d animation system. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1991), SIGGRAPH '91, ACM, pp. 113–122.

[42] Live2D Inc. What is live2d. `https://www.live2d.com/en/about/`, 2023. Accessed: 2023-08-07.

[43] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34*, 6 (Oct. 2015), 248:1–248:16.

[44] Lufkin, B. The virtual vloggers taking over youtube. *BBC Worklife* (Oct 2018).

[45] M2 Inc. Character animation tool e-mote. `https://emote.mtwo.co.jp/`, 2023. Accessed: 2023-08-07.

[46] Ma, Z., Zhu, X., Qi, G., Lei, Z., and Zhang, L. Otavatar: One-shot talking face avatar with controllable tri-plane rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

[47] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space, 2019.

[48] Mikatsuki Arpeggio. Mikatsuki Arpeggio. `http://roughsketch.en-grey.com/`, 2023. Accessed: 2023-09-14.

[49] Mikatsuki Arpeggio. Mikatsuki Arpeggio, Koakuma Mei. `http://roughsketch.en-grey.com/Entry/83/`, 2023. Accessed: 2023-09-14.

[50] Mikatsuki Arpeggio. Mikatsuki Arpeggio, Marietta. `http://roughsketch.en-grey.com/Entry/67/`, 2023. Accessed: 2023-09-14.

[51] Mikatsuki Arpeggio. Mikatsuki Arpeggio, Taoist Boy. `http://roughsketch.en-grey.com/Entry/110/`, 2023. Accessed: 2023-09-14.

[52] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[53] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *CoRR abs/1411.1784* (2014).

[54] Nagano, K., Seo, J., Xing, J., Wei, L., Li, Z., Saito, S., Agarwal, A., Fursund, J., and Li, H. Pagan: Real-time avatars using dynamic textures. *ACM Trans. Graph. 37*, 6 (dec 2018).

[55] Osman, A. A. A., Bolkart, T., and Black, M. J. STAR: A sparse trained articulated human body regressor. In *European Conference on Computer Vision (ECCV)* (2020), pp. 598–613.

[56] Park, E., Yang, J., Yumer, E., Ceylan, D., and Berg, A. C. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[57] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation, 2019.

[58] Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A. A. A., Tzionas, D., and Black, M. J. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019).

[59] Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., and Bao, H. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV* (2021).

[60] Preechakul, K., Chatthee, N., Wizadwongsa, S., and Suwajanakorn, S. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[61] Pumarola, A., Agudo, A., Martinez, A., Sanfeliu, A., and Moreno-Noguer, F. Ganimation: One-shot anatomically consistent facial animation. In *International Journal of Computer Vision (IJCV)* (2019).

[62] Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[63] Ren, Y., Li, G., Chen, Y., Li, T. H., and Liu, S. Pirenderer: Controllable portrait image generation via semantic neural rendering. In *ICCV* (2021).

[64] Roich, D., Mokady, R., Bermano, A. H., and Cohen-Or, D. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.* (2021).

[65] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), vol. 9351 of *LNCS*, Springer, pp. 234–241. (available on arXiv:1505.04597 [cs.CV]).

[66] Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings* (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery.

[67] Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., and Baraniuk, R. G. Wire: Wavelet implicit neural representations. In *CVPR* (2023).

[68] Saragih, J. M., Lucey, S., and Cohn, J. F. Real-time avatar animation from a single image. In *2011 IEEE International Conference on Automatic Face Gesture Recognition (FG)* (2011), pp. 213–220.

[69] Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A. Graf: Generative radiance fields for 3d-aware image synthesis, 2021.

[70] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)* (2015).

[71] Singh, R., Shukla, A., and Turaga, P. Polynomial implicit neural representations for large diverse datasets. In *CVPR* (2023).

[72] Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS* (2020).

[73] SSS LLC. Touhoku Zunko · Zundamon PJ Official HP. https://zunko.jp/, 2023. Accessed: 2023-10-03.

[74] SSS LLC. zzm_a1zunko11.png. https://zunko.jp/sozai/zunkot_s/zzm_a1zunko11.png, 2023. Accessed: 2023-10-03.

[75] Stanley, K. O. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*, 2 (jun 2007), 131–162.

[76] Sun, J., Wang, X., Wang, L., Li, X., Zhang, Y., Zhang, H., and Liu, Y. Next3d: Generative neural texture rasterization for 3d-aware head avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 20991–21002.

[77] Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS* (2020).

[78] Tewari, A., Elgharib, M., Bharaj, G., Bernard, F., Seidel, H.-P., Pérez, P., Zöllhofer, M., and Theobalt, C. Stylerig: Rigging stylegan for 3d control over portrait images, cvpr 2020. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (june 2020), IEEE.

[79] Tewari, A., Elgharib, M., BR, M., Bernard, F., Seidel, H.-P., Pérez, P., Zöllhofer, M., and Theobalt, C. Pie: Portrait image embedding for semantic control. vol. 39.

[80] Transpchan. Collaborative neural rendering using anime character sheets. https://github.com/transpchan/transpchan.github.io/blob/57efe17cdce35cf2c49c8d11ebd9bac108d1ac59/live3d/CoNR.pdf, 2022. Accessed: 2023-08-07.

[81] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[82] Ververas, E., and Zafeiriou, S. Slidergan: Synthesizing expressive face images by sliding 3d blendshape parameters. *International Journal of Computer Vision* (2020), 1–22.

[83] Wang, T.-C., Mallya, A., and Liu, M.-Y. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2021).

[84] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612.

[85] Wu, C. H., and la Torre, F. D. Unifying diffusion models' latent space, with applications to cyclediffusion and guidance, 2022.

[86] Xu, H., Song, G., Jiang, Z., Zhang, J., Shi, Y., Liu, J., Ma, W., Feng, J., and Luo, L. Omniavatar: Geometry-guided controllable 3d head synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

[87] Xu, Z., Zhou, Y., Kalogerakis, E., Landreth, C., and Singh, K. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics 39* (2020).

[88] Yang, Y., Sundaramoorthi, G., and Soatto, S. Self-occlusions and disocclusions in causal video object segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 4408–4416.

[89] Yenamandra, T., Tewari, A., Bernard, F., Seidel, H., Elgharib, M., Cremers, D., and Theobalt, C. i3dmm: Deep implicit 3d morphable model of human heads. In *Proceedings of the IEEE / CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021).

[90] Zhang, J., Xian, K., Liu, C., Chen, Y., Cao, Z., and Zhong, W. Cptnet: Cascade pose transform network for single image talking head animation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)* (November 2020).

[91] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[92] Zhang, W., Cun, X., Wang, X., Zhang, Y., Shen, X., Guo, Y., Shan, Y., and Wang, F. Sadtalker: Learning realistic 3d motion coefficients for stylized audio-driven single image talking face animation. In *CVPR* (2023).

[93] Zheng, M., Yang, H., Huang, D., and Chen, L. Imface: A nonlinear 3d morphable face model with implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20343–20352.

[94] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. View synthesis by appearance flow. In *European Conference on Computer Vision* (2016).

[95] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. View synthesis by appearance flow, 2017.

[96] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).

[97] Zhuang, Y., Zhu, H., Sun, X., and Cao, X. Mofanerf: Morphable facial neural radiance field. In *European Conference on Computer Vision* (2022).

A Full System's Architecture Details

A.1 U-Net with Attention

The new backbones of the half-resolution rotator and editor are U-Net with attention [23], which are frequently used in diffusion models. We base our architecture on conditional U-Nets in the diffusion autoencoder paper by Preechakul et al. [60]. There, the U-Net takes as input a feature tensor, a time value, and a 1D conditioning vector. The time value and the conditioning vector are mingled tensors derived from the input feature tensor through adaptive instance normalization (AdaIN) units [25] that are parts of residual blocks [20]. A residual block would have two AdaIN units that are applied in succession: the first for time and the second for conditioning vector. In the diffusion autoencoder paper, the conditioning vector is a 512-dimensional vector. In our case, the conditioning vector is the 6-dimensional pose vector.[6] For our networks, the time value is always 0 and is totally redundant. We kept the code related to time embedding in place in order to reduce implementation effort.

The configurations for the backbone networks are given in Table 8. Both networks scale down the feature tensors to 16×16 before scaling them back up to the original resolution. Attention blocks are only present at the 16×16 resolution. The bottleneck part of each network has 4 residual blocks alternating with three attention blocks. (In other words, there are $3 + 2 = 5$ attention blocks in total.) Each attention block has 8 attention heads.

[6] While the full pose vector has 45 parameters, only 6 that concern the movement of the body are relevant to the networks that we modify.

Hyperparameter	Half-resolution rotator	Editor
image resolution	256×256	512×512
# base channels	64	32
channel multipliers	1, 2, 4, 4, 4	1, 2, 4, 8, 8, 8
# residual blocks per level	1	1
# bottleneck residual blocks	4	4
resolution with attention blocks	16	16
# attention heads	8	8

Table 8: Configurations of the U-Net with attention backbones for the half-resolution rotator and editor.

The half-resolution rotator and the editor differ not only on the configurations of their backbones but how the backbones are "wrapped" by extra units to that they conform to the networks' interfaces. We discuss these extra units in the two following subsections.

A.2 Half-Resolution Rotator

The half-resolution rotator is depicted in Figure 8. It takes as input

1. $I^{\text{rest}}_{\text{half}}$, a $4 \times 256 \times 256$ RGBA image of the character in rest pose obtained by downscaling the original input image, and
2. $\mathbf{p}$, a 6-dimensional pose vector.

Because the U-Net with attention backbone takes in a $64 \times 256 \times 256$ tensor as input, $I^{\text{rest}}_{\text{half}}$ must be converted to this shape with a convolution layer. The backbone produces another $64 \times 256 \times 256$ feature tensor, which is then used to perform several image processing operations. (See Section 3.)

- *Warping.* The feature tensor is passed to a convolution layer to produce an appearance flow $I^{\text{flow}}_{\text{half}}$ of size $2 \times 256 \times 256$. It is then used to warp the input image ($I^{\text{rest}}_{\text{half}}$) to produce a warped image $I^{\text{warped}}_{\text{half}}$.
- *Direct generation.* The feature tensor is converted to a $4 \times 256 \times 256$ RGBA image, denoted by $I^{\text{direct}}_{\text{half}}$.
- *Blending.* The feature tensor is converted to a $1 \times 256 \times 256$ alpha map, which is then used to blend the warped image $I^{\text{warped}}_{\text{half}}$ and the directly generated image $I^{\text{direct}}_{\text{half}}$ together. The result is called $I^{\text{blended}}_{\text{half}}$.

The half-resolution rotator outputs $I^{\text{flow}}_{\text{half}}$, $I^{\text{warped}}_{\text{half}}$, $I^{\text{direct}}_{\text{half}}$, and $I^{\text{blended}}_{\text{half}}$. These image are used for training. However, at test time, $I^{\text{flow}}_{\text{half}}$ and $I^{\text{blended}}_{\text{half}}$ are used by the next network, the editor.

A.3 Editor

The architecture of the editor is depicted in Figure 9. It takes 4 inputs:

1. $I^{\text{rest}}_{\text{full}}$, the original character image at the 512×512 resolution,
2. $\mathbf{p}$, the 6-dimensional pose vector,
3. $I^{\text{blended}}_{\text{coarse}}$, which is $I^{\text{blended}}_{\text{half}}$ scaled up to 512×512, and
4. $I^{\text{flow}}_{\text{coarse}}$, which is $I^{\text{flow}}_{\text{half}}$ scaled up to 512×512.

The way the editor processes these input is quite similar to what the half-resolution rotator does. The pose vector is passed to the backbone directly. The input image $I^{\text{rest}}_{\text{full}}$ is convolved to create a $32 \times 512 \times 512$ tensor. The two other inputs are passed through what we call the "coarse input processing submodule," which carries out the following steps.

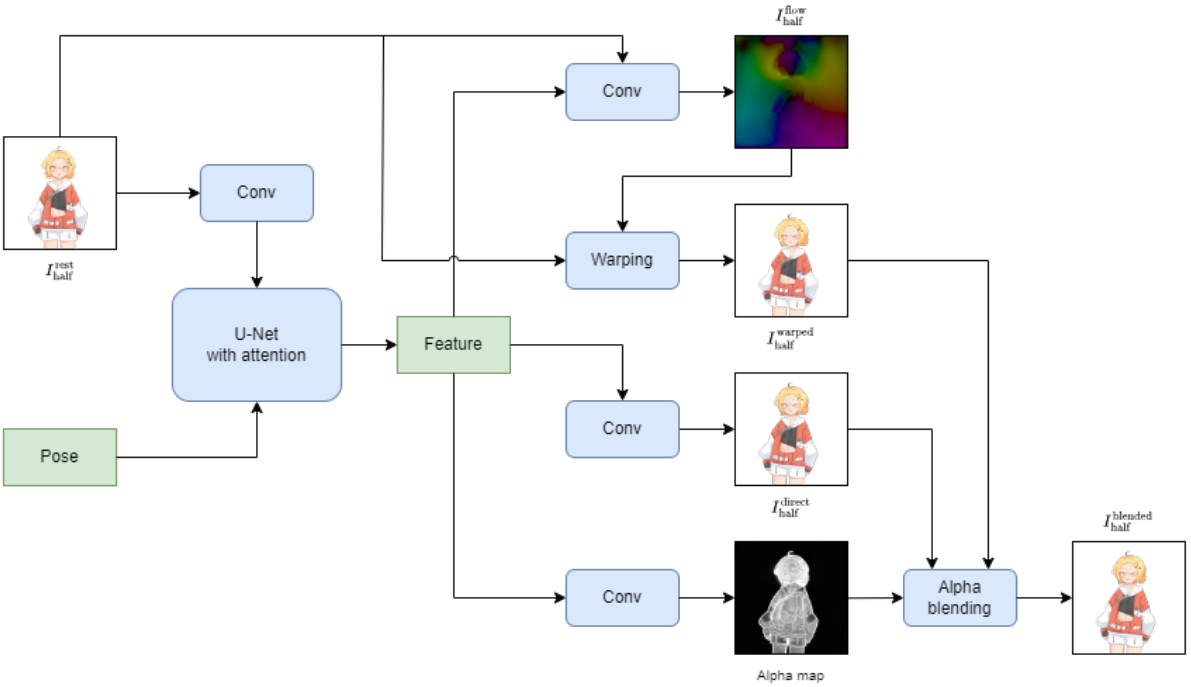

Figure 8: The new half-resolution rotator.

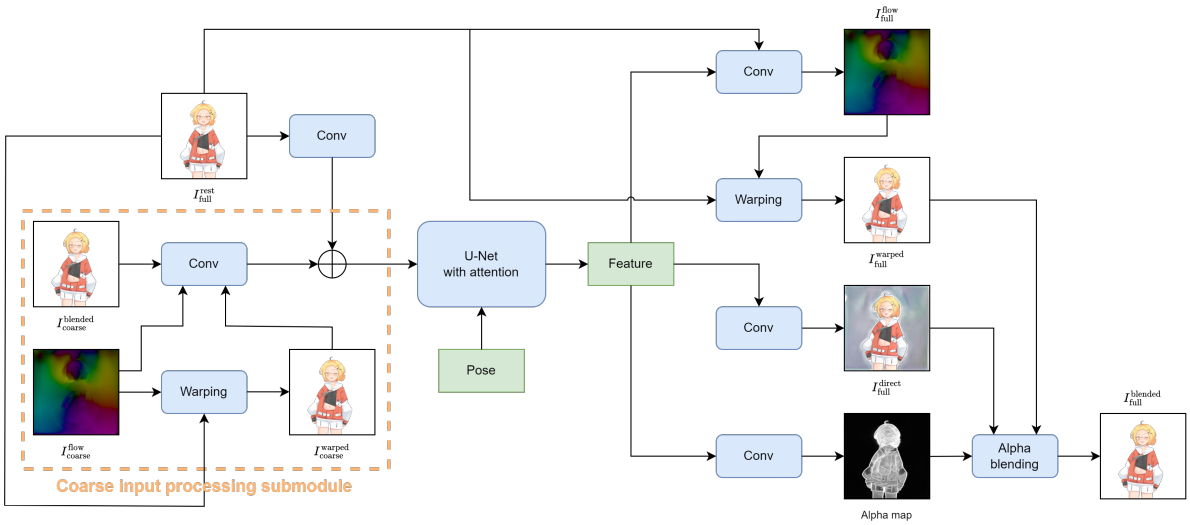

Figure 9: The new editor.

- First, the coarse appearance flow $I^{\text{flow}}_{\text{coarse}}$ is used to warped the original input image $I^{\text{rest}}_{\text{full}}$ to obtain the coarse warped image $I^{\text{warped}}_{\text{coarse}}$.
- Second, $I^{\text{blended}}_{\text{coarse}}$, $I^{\text{flow}}_{\text{coarse}}$, and $I^{\text{warped}}_{\text{coarse}}$ are concatenated, and the resulting tensor is convolved to form a $32 \times 512 \times 512$ tensor.
- Third, the result form the last step is added to the output of convolution layer that was applied to the original image to produce a $32 \times 512 \times 512$ tensor.

The resulting tensor is then passed to the backbone U-Net with attention. The output of the backbone is processed in the same way as what the half-resolution rotator does. This produces four tensors at full resolution: $I^{\text{flow}}_{\text{full}}$, $I^{\text{warped}}_{\text{full}}$, $I^{\text{direct}}_{\text{full}}$, and $I^{\text{blended}}_{\text{full}}$.

Note that, if we remove the coarse input processing submodule, the architecture of the editor would be exactly the same as the half-resolution rotator. Hence, the editor can be thought of as a network that also rotates the body given in the original image $I^{\text{rest}}_{\text{full}}$, but it takes the coarse inputs, $I^{\text{blended}}_{\text{coarse}}$ and $I^{\text{flow}}_{\text{coarse}}$, as hints. We will exploit this property in the training process of the editor.

B Full System's Training Details

B.1 Half-Resolution Rotator

The half-resolution rotator is trained with the following 6-termed loss that is a combination of the L1 loss and the perceptual content loss [29]. More concretely,

$$\mathcal{L}_{\text{HRR}} = \ell_{\text{L1}}\Big(\mathtt{L}^{\text{warped}}_{\text{L1}} + \mathtt{L}^{\text{direct}}_{\text{L1}} + \mathtt{L}^{\text{blended}}_{\text{L1}}\Big) + \ell_{\text{percept}}\Big(\mathtt{L}^{\text{warped}}_{\text{percept}} + \mathtt{L}^{\text{direct}}_{\text{percept}} + \mathtt{L}^{\text{blended}}_{\text{percept}}\Big).$$

The ℓ_{L1} and ℓ_{percept} are loss weights, which change once during the training process. (More on this later.) The loss terms that have subscripts "L1" are given by

$$\mathtt{L}^{\square}_{L1} = \frac{\|I^{\square}_{\text{half}} - I^{\text{posed}}_{\text{half}}\|_1}{C \times H \times W}$$

where C, H, and W are the channels, height, and width of the tensors, respectively. The $I^{\text{posed}}_{\text{half}}$ is the groundtruth posed image in the training dataset scaled down to 256×256, and $I^{\square}_{\text{half}}$ are the outputs of the half-resolution rotator as defined in Section A.2. The loss with subscripts "percept" is given by

$$\mathtt{L}^{\square}_{\text{percept}} = \Phi(I^{\square}_{\text{half}} - I^{\text{posed}}_{\text{half}})$$

and

$$\Phi(I_1, I_2) = \sum_{i=1}^{3} c_i\big(\|\phi_i(I_1^{\text{rgb}}) - \phi_i(I_2^{\text{rgb}})\|_1 + \|\phi_i(I_1^{\text{aaa}}) - \phi_i(I_2^{\text{aaa}})\|_1\big).$$

Here,

- $\phi_i(\cdot)$ denote the feature tensor outputted by the ith used layer in the VGG16 network [70], and we use the `relu1_2`,relu2_2, and `relu3_3` layers.
- c_i is the reciprocal of the number of components of $\phi_i(\cdot)$.
- I^{rgb} denotes the 3-channel image formed by dropping the alpha channel of image I.
- I^{aaa} denotes the 3-channel image formed by repeating the alpha channel of I three times.

We compute the perceptual loss as two L1 loss terms because the VGG16 network accepts an RGB image as input whereas the images outputted by the half-resolution rotator have 4 channels. To speed up the computation of $\Phi(\cdot,\cdot)$, we evaluate it stochastically. We flip a coin with head probability of 3/4. If it turns up head, we evaluate the term with I^{rgb}; otherwise, we evaluate the term with I^{aaa}. Of course, the terms are scaled with the reciprocal of the probability to make sure that the expectation has the right value.

Training is divided into two phases.

- In the first phase, only the L1 losses are used. In other words, $\ell_{\text{L1}} = 1$, and $\ell_{\text{percept}} = 0$. The first phase lasts for 1 epoch (500K training examples).
- In the second phase, all loss terms are used. In particular, we set $\ell_{\text{L1}} = 1$, and $\ell_{\text{percept}} = 0.2$. The second phase lasts for 18 epochs (9M training examples).

We used the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate starts at 0 and linearly increases to 10^{-4} over the first 5,000 training examples. The batch size was 16.

B.2 Editor

Training has three phases. In the first two phases, the coarse input processing submodule is dropped from the editor, making it temporarily a "full-resolution rotator." The network is trained using the training process of the half-resolution rotator but now with the full resolution images instead of the half-resolution ones.

In the third phase, we added the coarse input processing submodule back and train the network to minimize the following loss:

$$\mathcal{L}_{\text{ED}} = \lambda_{\text{L1}}\Big(\mathcal{L}_{\text{L1}}^{\text{warped}} + \mathcal{L}_{\text{L1}}^{\text{direct}} + \mathcal{L}_{\text{L1}}^{\text{blended}}\Big) + \lambda_{\text{percept}}\Big(\mathcal{L}_{\text{half}}^{\text{direct}} + \mathcal{L}_{\text{half}}^{\text{blended}} + \mathcal{L}_{\text{quad}}^{\text{direct}} + \mathcal{L}_{\text{quad}}^{\text{blended}}\Big).$$

We fixed $\lambda_{\text{L1}} = 1$ and $\lambda_{\text{percept}} = 0.2$. The L1 losses are given by

$$\mathcal{L}_{\text{L1}}^{\square} = \|I_{\text{full}}^{\square} - I_{\text{full}}^{\text{posed}}\|_1$$

where $I_{\text{full}}^{\text{posed}}$ is the groundtruth posed image from the training dataset, and the $I_{\text{full}}^{\square}$ are the outputs of the editor as defined in Section A.3. The losses $\mathcal{L}_{\text{half}}^{\text{direct}}$, $\mathcal{L}_{\text{half}}^{\text{blended}}$, $\mathcal{L}_{\text{quad}}^{\text{direct}}$, and $\mathcal{L}_{\text{quad}}^{\text{blended}}$ are perceptual losses. The superscripts indicate the outputs of the editor that we compute the losses with, and the subscripts indicate how the losses are computed. The "half" subscript indicates that the images are scaled down to 256×256:

$$\mathcal{L}_{\text{half}}^{\square} = \Phi\Big(\textsc{Down}(I_{\text{full}}^{\square}), \textsc{Down}(I_{\text{full}}^{\text{posed}})\Big)$$

where $\textsc{Down}(\cdot)$ denotes scaling a 512×512 image down to 256×256. The "quad" subscript indicates that the images are divided into four quadrants so that a 512×512 images becomes four $4 \times 256 \times 256$ images. The quadrants are then used to evaluate the perceptual losses.

$$\mathcal{L}_{\text{quad}}^{\square} = \sum_{i=1}^{4} \Phi\Big(Q_i(I_{\text{full}}^{\square}), Q_i(I_{\text{full}}^{\text{posed}})\Big)$$

where $Q_i(\cdot)$ extracts the ith quadrant from the argument. We found that evaluating the perceptual losses at 256×256 rather than 512×512 led to a network that produced sharper images.

Again, we use the same optimizers and learning rate schedule as those of the half-resolution rotator. The first phase lasts for 1 epoch (500K examples), the second 18 epochs (9M examples), and the third 18 epochs (9M examples).

(a) Character face (b) Mask

Figure 10: Binary mask that covers movable facial parts of a character.

C Computers Used for Speed Measurements

We conducted experiments that measured time it took for the models to produce a single animation frame on the following 3 desktop computers.

- **Computer A** has two Nvidia Nvidia RTX A6000 GPUs, a 2.10 GHz Intel Xeon Silver 4310 CPU, and 128 GB of RAM. It represents a computer used primarily for machine learning research.
- **Computer B** has an Nvidia Titan RTX GPU, a 3.60 GHz Intel Core i9-9900KF CPU, and 64 GB of RAM. It represents a high-end gaming PC.
- **Computer C** has an Nvidia GeForce GTX 1080 Ti GPU, a 3.70 GHz Intel Core i7-8700K CPU, and 32 GB of RAM. It represents a typical (yet somewhat outdated) gaming PC.

D Student Face Morpher's Loss Function

The student face morpher is trained to minimize the following loss:

$$\mathcal{L}_{\text{fm}} = E_{\mathbf{p}\sim p_{\text{pose}}}\Big[\big\|S^{\text{fm}}(I_{\text{in}},\mathbf{p}) - T^{\text{fm}}(I_{\text{in}},\mathbf{p})\big\|_1 + \lambda_{\text{fm}}\big\|M \odot (S^{\text{fm}}(I_{\text{in}},\mathbf{p}) - T^{\text{fm}}(I_{\text{in}},\mathbf{p}))\big\|_1\Big].$$

Here,

- I_{in} is the image of the character that we want to create a specialized student model of.
- $\mathbf{p}$ is a pose vector, which is sampled from the training dataset of the full system.
- p_{pose} is the uniform distribution over the poses in the training dataset.
- $S^{fm}(\cdot,\cdot)$ is the student face morpher.
- $T^{fm}(\cdot,\cdot)$ is the teacher face morpher, which comes from the full system in Section 3.
- M is a binary mask that covers all the movable facial organs: eyebrows, eyes, mouth, and chin. This mask has to be created for each individual character. See Figure 10 for an example.
- Lastly, λ_{fm} is a weighting constant, whose value is 20 in all experiments.