Scaling Neural Face Synthesis to High FPS and Low Latency by Neural Caching

Frank Yu
Sid Fels
Helge Rhodin
University of British Columbia (UBC)
frankyu@cs.ubc.ca
ssfels@ece.ubc.ca
rhodin@cs.ubc.ca

Abstract

Recent neural rendering approaches greatly improve image quality, reaching near photorealism. However, the underlying neural networks have high runtime, precluding telepresence and virtual reality applications that require high resolution at low latency. The sequential dependency of layers in deep networks makes their optimization difficult. We break this dependency by caching information from the previous frame to speed up the processing of the current one with an implicit warp. The warping with a shallow network reduces latency and the caching operations can further be parallelized to improve the frame rate. In contrast to existing temporal neural networks, ours is tailored for the task of rendering novel views of faces by conditioning on the change of the underlying surface mesh. We test the approach on view-dependent rendering of 3D portrait avatars, as needed for telepresence, on established benchmark sequences. Warping reduces latency by 70% (from 49.4ms to 14.9ms on commodity GPUs) and scales frame rates accordingly over multiple GPUs while reducing image quality by only 1%, making it suitable as part of end-to-end view-dependent 3D teleconferencing applications.

1. Introduction

Telepresence via photo-realistic 3D avatars promises to better connect people. Recent advances in neural rendering already enable near photo-realistic image quality, but the underlying deep neural networks limit the best possible latency with their sequential, layer-wise processing. This is a problem for virtual reality applications as these require low latency upon head motion of the user to convey an immersive experience. For example, [33] creates a high-fidelity system using 120Hz projectors and user viewpoint tracking with a tracker having 60Hz updates and 3ms latency to minimize users' perception of warping of the scene when they move. However, none of the existing neural renderers reaches the required *motion to photons latency*, i.e., the time it takes from the user input, such as moving the head in VR, until the display updates. It is henceforth an open problem to find improved neural models that strike a better trade-off between speed and image quality.

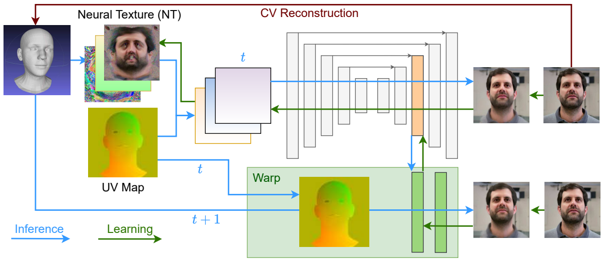

Figure 1: **Neural Rendering with Warping.** While a single frame method requires a deep neural network to synthesize a realistic head image from rough geometry, our implicit warping yields low latency for a new frame at $t + 1$ with a shallow network.

We develop a parallel implementation, inspired by previous approaches for improving the frame rate for other video processing tasks [3] by spreading the computation over several frames over multiple GPUs. By itself, parallel execution only improves frame rate, not the latency that is so crucial for VR, since the effective network depth of sequentially executed layers remains the same. To nevertheless reach the desired latency reduction and frame rate, we combine parallel execution with a dedicated warp layer that is tailored for neural face synthesis and acts as a skip connection between consecutive frames.

This neural caching approach re-uses information from the previous time step to improve latency from the current one with a shallow network for image generation and a deep network for computing the cache while waiting for the next input frame. Note that this caching strategy by itself already improves latency on a single GPU and scales in frame rate on increasingly parallel hardware by breaking the sequential layer dependencies and offloading the image cache generation into multiple threads.

The difficulty lies in finding the right representation for the cache to succeed with a low-capacity network. Our approach is inspired by classical low-latency rendering on VR displays [27], which compute only the position of objects for in-between frames and warp color from the previous. However, explicit warping does not apply well to neural renderers where the underlying geometry is approximate and the neural texture is high-dimensional making warping operations much more costly.

To warp the neural representation, we introduce an implicit warp that provides a skip connection that is tailored for neural rendering by taking into account the head model parameterization. The result is a warp network that models the image-space motion from one frame to the next given the desired viewpoint and head model parameters. Figure 1 outlines the main components.

Our design is geared towards *novel-view-synthesis* of a talking head given a dynamically changing viewpoint, such as the user's head motion in VR. We build upon the deferred neural renderer (DNR) [24] that uses a neural texture learned at training time. Our contributions towards scaling frame rate and latency on parallel hardware are:

- Demonstrating that the proposed neural caching can reduce latency by up to 70% with minimal degradation in image quality (only 1% PSNR).
- Extending a DNR to generate a high-resolution output with low latency via caching and an implicit warping.
- Developing a parallel scheduler that supports warping and synchronizes multiple threads using queues.
- Refining the representation of facial expression, head pose, and camera angle to improve the implicit warp.
- Adding head-stabilization and tweaking the backbone and training strategy for noisy real-world conditions.

Our experiments highlight the importance of how to cache as well as what and how information from the current frame is passed to the shallow *implicit warping network*. Figure 2 compares the most related methods. Ours strikes the best latency and FPS improvement with the least image quality trade-off.

2. Related Work

In this section we discuss recent high-quality image generation methods and contrast with those that optimize runtime with a focus on face generation.

Photo-realistic synthesis. High-quality photorealistic rendering is booming, using either implicit scene representations such as Neural Radiance Fields (NeRFs) [19] or deep neural networks trained on GAN objectives [14, 15], which can also be conditioned on viewpoint and pose changes [4, 23]. However, these implicit models all rely on very deep neural networks that do not run at high-enough frame rates or are limited to static scenes via pre-computed acceleration structures [31].

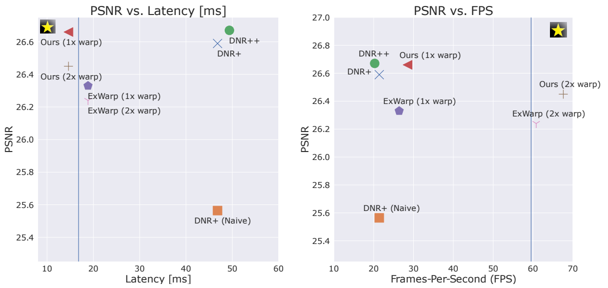

Figure 2: **Performance vs. latency (left) and fps (right)** for our models and the baselines on the *beard* dataset. Optimal performance is in the top-left and top-right corners, indicated by the yellow star in each respective figure. The vertical blue line marks the latency and fps of [32]. Models suitable for parallel execution are run using two GPUs.

Talking head models. For dynamic faces, it is most common to start from a parametric face model that parameterizes identity and expression in the form of blend shapes that are linearly combined with the full mesh [1, 17]. The coefficients of these models are low-dimensional and therefore suitable for driving avatars in telepresence and for reenactment by mixing identity and pose information of two subjects. While earlier models focus on only modeling the facial region [29, 25, 26, 22, 24] conditioning on a complete head model, such as FLAME [17], enables novel view synthesis of side views [9] suitable for view-dependent 3D systems[35]. We follow this line of work and extend it by reducing the rendering latency for interactive applications.

Another line of work developed subject-agnostic face synthesis [21, 20, 8, 7] that is conditioned on single or multiple images instead of a subject-specific 3D model. However, this comes at the price of reduced details and entanglement of expressions and head motion [30], particularly if the reference and target poses differ largely, and may require explicit warping operations that are relatively costly and ill-defined for occluded regions [32]. Although [32] is already extremely fast on medium image resolution with 4ms runtime, there is still room for improvement when rendering high-resolution images and medium resolution on embedded devices.

Caching approaches. Re-using information from the previous time step has been used for many computer vision tasks, among them: object detection [18], video action recognition [5], and segmentation [10]. Carreira et al. [3] gives an excellent overview of different architectures for video processing, including: *i) depth parallel* architectures that execute a deep neural network over several iterations, leading to a delay equal to the depth of the layers; *ii) depth parallel + skip* in which the head (last couple of network layers) of a depth parallel network is updated with the new input via a skip connection; and, *iii) multi-rate clock* architectures in which the input features for the head are not updated at every time step and the head and backbone operate at different clock rates. Our caching approach follows the multi-rate clock pattern. However, none of the existing parallel models has demonstrated image generation. Our contribution is to tailor this general concept to the problem of novel-view-synthesis of faces via neural rendering by a suitable form of warping.

3. Preliminaries

Our goal is an efficient neural renderer that outputs an image $\mathbf{I} \in \mathbb{R}^{H \times W}$ of a face parameterized as a surface mesh with vertices $\mathbf{v} \in \mathbb{R}^{3 \times K}$, triangle indices $\mathbf{i} \in \mathbb{R}^{3 \times K}$, and associated texture coordinates $\mathbf{u} \in \mathbb{R}^{2 \times K}$ and neural texture $\mathbf{N} \in \mathbb{R}^{D \times H \times W}$ [24]. These are the same inputs that a normal renderer would expect, except that the texture $\mathbf{N}$ is D-dimensional instead of storing three color values.

Deferred neural rendering. Our starting point is the deferred neural renderer introduced by Thies et al. [24], which approximates the complex and computationally expensive rendering equation with a convolutional neural network G. Figure 1 gives an overview including the differences of [24] applied to our full model including caching. Initially, a rasterizer renders UV maps $\mathbf{U} \in \mathbb{R}^{2 \times H \times W}$ of the textured mesh. These are of the same dimension as the output image and store for every pixel the corresponding texture coordinate. Sampling these locations from $\mathbf{N}$ gives the feature map $\mathbf{F} \in \mathbb{R}^{D \times H \times W}$. For classical deferred rendering, we would sample from a color texture and combine it with light position information to form the final image. In the case of the neural renderer, the texture has more than three channels forming learnable features. The network G turns $\mathbf{F}$ into the final image, replacing geometric illumination computations and material shaders in classical rendering. This forward pass is traced with blue arrows in Figure 1 while the backwards information flow during training is marked in green.

Training and face reconstruction. The parameters of the involved neural network G as well as the neural texture itself are trained on a large dataset that has examples of the input 3D mesh and a high resolution image of the face—the desired output. We use real videos of the person as input and reconstruct vertices $\mathbf{v} \in \mathbb{R}^{3 \times K}$, expression coefficients $\mathbf{e} \in \mathbb{R}^{50}$, PCA shape parameters $\mathbf{s} \in \mathbb{R}^{100}$, and head pose $\theta \in \mathbb{R}^{6}$ of the FLAME parametric model [17] alongside camera position $\mathbf{p} \in \mathbb{R}^{3}$ using the off-the-shelf estimator DECA [6]. Internally, it is using the 2D keypoint detector from [2]. The reconstructed face overlays well with the image when re-projected, but details, such as hair and ears are often misaligned, which puts a larger burden on the neural renderer to synthesize these. This reconstruction step is marked with a red arrow in Figure 1.

The generator G is a U-Net and the neural texture $\mathbf{N}$ is initialized at random and subsequently optimized by backpropagation to store details of the training object locally. The loss function is the L1 difference between the rendered and the reference image in the dataset. This backwards pass is traced with green arrows in Figure 1. Training is on cropped images, which speeds up training.

Base architecture. We use the 10-layer U-Net as in [11] and a multi-scale neural texture with four levels of detail as in [24]. Furthermore, to better model viewpoint-dependent effects, the view direction is projected to 9 spherical harmonics coefficients which are subsequently multiplied to channels 4 through 12 of the feature map. This enables explicit encoding of view-dependent effects similar to positional encoding [28].

Real-time viewpoint-dependent rendering. Our primary application fields is 3D teleconferencing, where a person must be rendered at a high frame rate, with low latency, from the viewpoint of the user that is roughly frontal. Given a new view direction, e.g., from an eye-tracker, our focus is on generating a natural looking image of this novel view as quickly as possible to mitigate motion sickness, reduce warping artifacts [34], and avoid discomfort. The object motion capture could be performed offline or through a slower channel, as usually only the viewpoint-dependent rendering demands the low-latency. In telepresence applications, bandwidth is dictated by the size of the FLAME model; estimated on the source side, transferred, and rendered by our system given a new view from the receiver.

4. Method

In this section, we introduce our neural caching approach, and propose two variants that operate on single and multi-GPU systems and are respectively tuned for latency and frame rate. We cache information from the immediately preceding frames. Thereby, the motion that must be bridged from the cached information to the current frame is small, which allows us to introduce an implicit warp that attains maximum performance.

Neural Cache. For our neural caching, we first run the deep and slow image generator $G(\mathbf{N}_t, \mathbf{U}_t, \mathbf{p}_t)$ that is conditioned on the neural texture $\mathbf{N}$, UV-map $\mathbf{U}_t$ and view direction (e.g., user's head motion in VR) of the current frame t. Figure 3 provides a detailed overview of our pipeline. We cache features $\mathbf{C}_t^{(3)}$, $\mathbf{C}_t^{(4)}$, and $\mathbf{C}_t^{(5)}$ from the last 3 layers of the generator together with camera position $\mathbf{p}_t$, and spherical harmonics (SH) encoding of the pose, $\mathbf{h}_t^{\text{obj}}$. Furthermore, we add the UV map $\mathbf{U}_t$, expression $\mathbf{e}_t$, and the pose θ_t that are specific to rendering faces. Formally we write the combined cache $\mathbf{C}$ as

$$\mathbf{C}_t := [\mathbf{C}_t^{(3)}, \mathbf{C}_t^{(4)}, \mathbf{C}_t^{(5)}, \theta_t, \mathbf{p}_t, \mathbf{e}_t, \mathbf{h}_t^{\text{obj}}, \mathbf{U}_t]. \qquad (1)$$

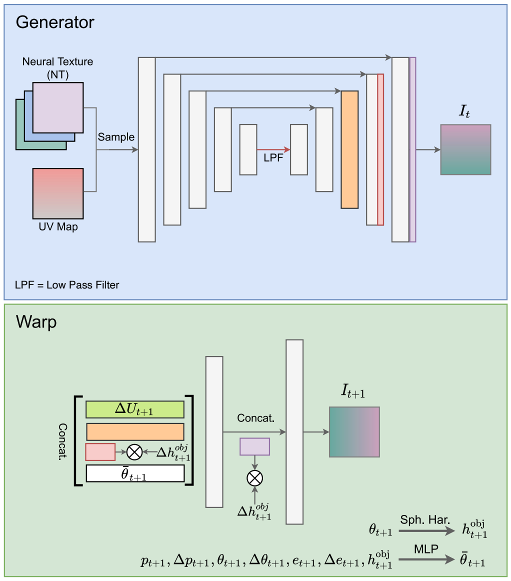
Generator
Neural Texture (NT)
Sample
UV Map
LPF
I_t
LPF = Low Pass Filter
Warp
Concat.
ΔU_{t+1}
Δh^{obj}_{t+1}
$\bar{\theta}_{t+1}$
Concat.
Δh^{obj}_{t+1}
I_{t+1}
θ_{t+1} Sph. Har. h^{obj}_{t+1}
$p_{t+1}, \Delta p_{t+1}, \theta_{t+1}, \Delta\theta_{t+1}, e_{t+1}, \Delta e_{t+1}, h^{\text{obj}}_{t+1}$ MLP $\bar{\theta}_{t+1}$

Figure 3: **Our low latency pipeline** caches the previous frame (features from the generator (left) marked orange and red) and the pose of the person to generate the subsequent frames with only two additional up-convolution layers (right) as soon as a new view direction is available. The generator and warping networks are learned end-to-end and can be applied in parallel at inference time.

Implicit Warping. Previous work used an explicit warp operation from a reference frame, which requires preceding neural network layers to predict the warp and is implemented as a texture sampling step that is relatively slow due to random access patterns for every pixel. We propose an implicit warp with a neural network W that is composed of only two up-convolution layers. These two layers take in cached information $\mathbf{C}_t$, rendered UV map and the new camera, pose, and expression from the new frame $t+1$, to reconstruct the image I_{t+1}. This network is intentionally kept shallow to decrease the latency of image generation.

In addition, we found in a detailed ablation study that giving as input the new camera position $\mathbf{p}_{t+1}$, object pose θ_{t+1}, expression $\mathbf{e}_{t+1}$ and their differences to the previous frame, via a single-layer MLP, M, works best. This yields

$$\bar{\theta}_{t+1} = M(\mathbf{p}_{t+1}, \Delta\mathbf{p}_{t+1}, \theta_{t+1}, \Delta\theta_{t+1}, \mathbf{e}_{t+1}\Delta, \mathbf{e}_{t+1}, \mathbf{h}^{\text{obj}}_{t+1}), \tag{2}$$

where the Δ refers to the change in a quantity between two frames, here $\theta_{t+1} - \theta_t$. Additionally, we also include the UV map $\mathbf{U}_{t+1}$. Together with the cached features processing by the first warping layer W_1 gives

$$\mathbf{F}_1 = W_1(\mathbf{C}_t^{(3)}, \mathbf{C}_t^{(4)}\Delta\mathbf{h}^{\text{obj}}_{t+1}, \bar{\theta}_{t+1}, \Delta\mathbf{U}_{t+1}) \tag{3}$$

where $\Delta\mathbf{h}_{t+1}$ and $\bar{\theta}_{t+1}$ are broadcasted to the resolution of the cache and $\Delta\mathbf{h}_{t+1}$ modulates the feature map by multiplication, similar to how positional encoding works, but here for rotational changes. These features are further processed to output the image with $I_{t+1} = W_2(\mathbf{F}_1, \mathbf{C}_t^{(5)}\Delta\mathbf{h}^{\text{obj}}_{t+1})$. Note that W warps the previous to the current frame, but without a spatial transformation layer—it does so implicitly via a local approximation of how the image changes with respect to changes in pose.

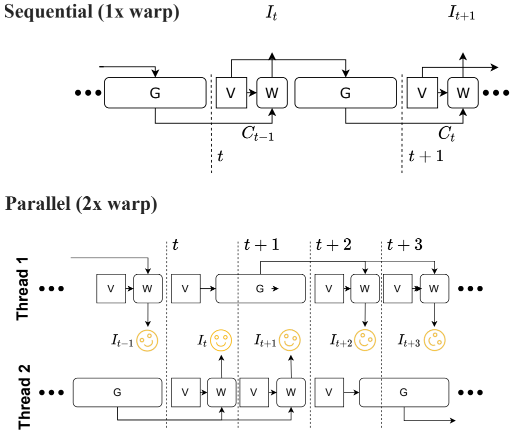

Figure 4: **Our rendering pipelines for sequential (Top) and parallel execution (Bottom).** A network W warps a cache from the previous frame as soon as a new viewpoint V is ready for rendering, producing the output image much quicker than the full generator G could. Leveraging two threads further increases the output frame rate.

4.1. Operation Modes and Parallel Execution

Our approach scales easily between single and parallel GPU execution. When only a single GPU is available, Figure 4-top visualizes the principle of reducing latency with a shallow warp network, compared to running the slower generator. Here, the cache is updated in sequence with the warp, thereby reducing latency but not frame rate.

When multiple GPUs are available, we can combine the proposed implicit warp with parallel execution, thereby rendering at a two or more times higher frame rate by warping a single or multiple images $\mathbf{I}_t, \mathbf{I}_{t+1}, \cdots$ while the generator runs on a separate GPU on a separate thread. Figure 4-right showcases this scheme. Notably, even though the warping is an approximation, multiple warping does not reduce image quality when operating online on high-frame rate video streams. The higher processing speed of warping twice in parallel reduces the distance between frames that can be processed and thereby makes the two-frame warp as difficult as a single-frame warp operating at half the frame rate.

We experimented with different job assignment and synchronization schemes between the two threads. We found that threads alternating between image generation and warping is most efficient and eases implementation. In this model, the main thread distributes newly arriving viewpoint information to the two worker threads via a queue, each associated to one GPU. These in turn wait for new data in the queue. Upon receiving new data, they alternate between caching and warping as visualized in Figure 4-bottom. Thereby, the warping is executed on the same GPU as the generator, such that the cache can remain on the same GPU. The alternative of using a dedicated warp and cache thread has a much lower performance since the cache would have to be moved from one GPU to CPU and then again from CPU to the target GPU. The required synchronization of ques has a negligible overhead in our implementation on two RTX 2080s with only 0.25ms/frame. Moreover, our preliminary attempt of using manual locks instead of queues was more complex without improving performance. We will make our implementation publicly available to facilitate further research.

4.2. Improved Neural Head Rendering

Our starting point, DNR [24], is a general rendering approach. In the following, we explain our architectural changes towards tuning it for face synthesis.

Head stabilization To reduce jitter and flicker of the head, we ensure that the virtual camera that we use to generate our UV masks is always centered on the subject's head and has consistent scale when generating videos by centering the camera on the head midpoint and scaling by the projected ear-to-ear distance. Because the driving motion capture signal is often unstable, we further smooth the global head position $\mathbf{p}_t$ with a delayed Gaussian filter of size five and fix the identity $\mathbf{s}$ of the FLAME model to be the mean identity estimated by DECA on the training set. To maintain facial expressions and lip motion faithfully, the jaw orientation in θ and expression parameters $\mathbf{e}$ are left untouched.

Loss Function For our baselines which only predict the image at time t we define the loss function as

$$L_{\text{train}} = \lambda_{\text{tex}} L_{\text{tex}} + \lambda_{\text{img}} L_{\text{img}} + \lambda_{\text{p}} L_{\text{p}}, \tag{4}$$

which measures the L1 distance between the first three channels of the sampled neural texture and the ground truth image (L_{tex}), the L1 photometric loss (L_{img}), and the perceptual loss [12] between the predicted image and ground truth image (L_p). We weight these loss terms with coefficients 1, 1, and 0.1. For the warping we add λ_{img} and L_{p} on the future frame $t+1$ and down-weight the existing L_{img} by 0.1 to put much more weight on the prediction for I_{t+1} that is used at inference time.

Architectural Changes. Based on the work of [13] we make the following improvements to the base DNR [24] network to improve output image quality. First, we replace the transpose convolutions in the latter half of of the U-Net with a bilinear upsample layer followed by a 2D convolution (up-convolution). This has been shown to increase the final image quality of output and reduce grid-like noise in reconstructed images. Furthermore, we apply a Gaussian low pass filter (LPF) on the smallest spacial features of the U-Net architecture. We refer to the baselines that utilize all these improvements as DNR+.

5. Experiments

We evaluate our Neural Warping technique with respect to our goal of maximizing image quality and minimizing the latency by applying our Neural Warping. Figure 2 summarizes our main results on the possible trade-offs between accuracy vs. latency and accuracy vs. fps, comparing our two variants to the most related work and showing an improvement in fps of up to 300% and a reduction in latency of 70%. We also provide an ablation study to identify specific trade-offs with respect to individual components of our warping network. The supplemental material contains additional results.

Baselines. DNR [24] is the backbone we use as our reference. **DNR+** improves DNR with recent neural network architecture improvements from [13]. We furthermore add **DNR++** that uses the current and past frame as input. DNR+ and DNR++ both act as a theoretical upper bound for our model's image quality and therefore, provide a good measure of the effectiveness and efficiency of our approach. We also compare against the recent method from **Wang et al.** [30], using their online interface (http://imaginaire.cc/vid2vid-cameo/) and in terms of runtime to **Zakharov et al.** [32] using the same 512 × 512 image resolution. We do not provide PSNR numbers as it was designed for a much smaller resolution of 256 × 256. In addition, we create a *naive* baseline where we shift input frames by one to emulate the delay that incures when running a large image generator without warping. It serves as an expected lower bound on accuracy.

Datasets. We use the talking head sequences from [24] to compare against prior work as well as a *beard* and a *high-fps* sequence with more difficult facial hair that was recorded with ethics board approval. Videos for the ***beard*** and ***high-fps*** dataset were recorded at 1920x1080 and split into, respectively, 2604/558/558 and 3600/500/1000 frames for training/validation/testing. ***Trump*** [24] sequences is a 1280x720 of 431 frames. The ***Obama*** [24] sequence is a 512x512 of 2412 frames. The ***male*** and ***female*** [24] sequences are both 768x768 with 2380 frames. The *high-fps* recording has 60 fps, all others run at 30 fps.

Training Setup. We train all our models for 150 epochs using Adam [16] with betas equal to {0.9, 0.999}, learning rate of 1e-4 for both the Generator, G and Warp network W, and 1e-3 for the neural texture.

Metrics. We evaluate our image reconstruction accuracy

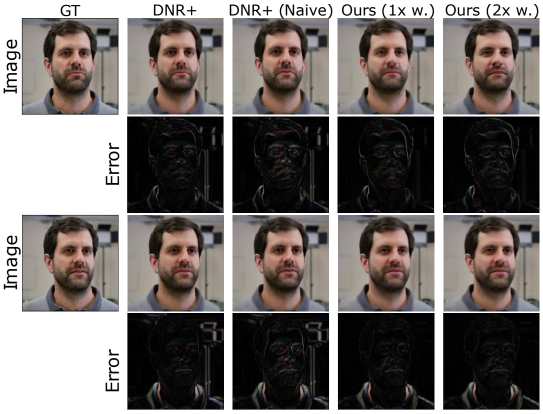

Figure 5: **Frames and their error maps** for our model (1x and 2x warp), DNR+, and naive baseline. Based on the L1 error we can see all models perform similarly, differences are visible in the error maps for high-frequency details.

using an L1 reconstruction error between the ground truth and reconstructed image as well as PSNR and SSIM. Furthermore, to show our increase in speed, we report the latency and frames-per-second (fps) of our models and baselines. For all methods, we only measure the time it takes to process the input UV maps and the corresponding skeleton (pose), expression, and camera (extrinsics) information; excluding the time it takes to render the UV map because they are implementation dependent and with negligible overhead when implemented in a rasterizer. Similarly, we only account for the time the inference generator takes in [32], ignoring the processing of the conditional keypoint image. All latency and fps metrics are computed on NVIDIA RTX 2080 GPUs. For our approaches we report metrics and timings when we cache every frame (1x warp) and every second frame (2x warp) to show our method is robust to varying deviations between input frames.

5.1. Latency and Runtime Improvement

The generator backbone has a runtime of 47.02ms and equivalent latency. Our warp net has a runtime of 14.62ms and therefore deduces latency by a factor of 3.2. Running on multiple GPUs improves frame rate from 28.5 to 67.6 while inducing only a negligible 0.25 ms increase in latency due to the required synchronization. Note that parallel execution across multiple GPUs can by itself not improve the latency of a sequential process. When processing even and odd frames on different GPUs, the time from input view to rendering output remains the same. The impact of different model configurations on latency is evaluated in Table 3.

5.2. Offline Reconstruction Quality

To test image generation quality, we use a held-out test video and drive the trained models using the FLAME head model reconstructed on the reference video. The results from Figure 5 and Table 1 evaluate the difference to the reference video in offline processing mode. When comparing individual frames between the baseline and our model, Figure 5 shows that single (1x warp) and multi-frame warping (2x warp) work nearly as well in terms of error. Table 1 reveals, somewhat surprisingly that 1x warp slightly outperforms the DNR+ baseline in terms of PSNR despite having to warp with a shallow network. This is possible since it has access to the current and past frame that helps to correct errors in the facial expression estimation. To this end, we introduced the DNR++ baseline as a new upper bound that also has access to the two previous frames. In summary, at a small reduction in accuracy compared to the baseline models, latency and frame rate are greatly improved by a factor of two or more.

5.3. Online Reconstruction Quality

Online reconstruction requires the algorithm to run at the native frame rate of the video. This has a significant influence on the performance of our algorithm as the warping operation becomes simpler for high-frame rate videos where the motion between two frames is reduced, leading to even higher performance gains than for the previous offline evaluation. We test this effect on the *high-fps* sequence shown in Fig. 6. Table 2 reveals that running online on 60 fps videos with Ours (2x warp) improves on Ours (1x warp), as the latter can only process every other frame requiring larger warps. Hence, warping multiple times is beneficial, jointly improving in latency, runtime, and image quality, when parallel hardware is available. The basic DNR baselines do not even run at 30 fps and are hence not comparable in the high-fps online setting.

Note that absolute PSNR numbers differ across subjects and scenes since faces are smaller/bigger and also contain

Model	#GPU	L1 ↓	PSNR ↑	SSIM ↑	Latency [ms] ↓	FPS ↑
timing baseline [32]	1	-	-	-	16.60	60.2
DNR+ (Naive)	1	0.0278	25.56	0.8970	46.81	21.4
DNR+	1	0.0240	26.59	0.9165	46.81	21.4
DNR++	1	**0.0237**	**26.67**	**0.9168**	49.37	20.3
ExWarp as in [32] (1x warp)	2	0.0257	26.33	0.9108	18.84	26.4
ExWarp as in [32] (2x warp)	2	0.0260	26.24	0.9094	18.84	60.8
Ours (1x warp)	1	0.0244	26.66	0.9107	**14.62**	16.3
Ours (1x warp)	2	0.0244	26.66	0.9107	14.87	28.5
Ours (2x warp)	1	0.0251	26.45	0.9069	**14.62**	26.3
Ours (2x warp)	2	0.0251	26.45	0.9069	14.87	**67.6**

Table 1: **Offline evaluation on the *beard* dataset.** As expected, our model does not achieve the best metrics on the image reconstruction metrics, but they outperform the baselines in terms of latency and fps. Timing results for *Ours 2x warp* are using parallel execution.

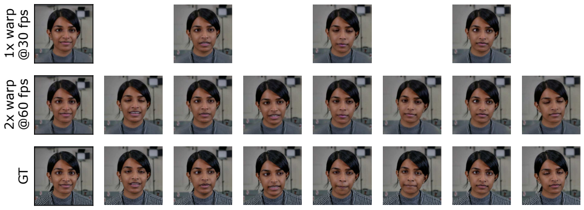

Figure 6: **Online application.** When running at their native frame rate, high-fps models (2x warp) improve as they have to bridge a smaller gap between frames.

more or less high-frequency details, including the texture on the shoulder region that is of little concern for the facial feature reconstruction fidelity. Hence, it does not make sense to relate absolute but only relative numbers between the *beard* and *high-fps* scores.

5.4. Novel View Synthesis Quality

Reproducing a pre-recorded sequence does not necessarily need low latency since the entire video could be cached. Yet, latency is crucial for rendering a face from a novel viewpoint to account for the user's head motion in VR and in general for viewpoint-dependent displays. We generate novel views of characters by rotating their underlying 3D mesh (which is used to generate our input UV maps) while holding other parameters fixed. Figure 8 shows the retargeting of poses and views across videos and 7 shows synthetically generated views and compares it to the results from [30]. Because we condition on a full 3D face model, our rotation is more precise and keeps the pose unchanged compared to the learned 3D features from Wang et al. that lead to opening of the mouth and up-rotation. The quality by [30] is expected to be slightly lower as it is not person-specific, which serves our motivation to train a person-specific model.

5.5. Retargeting

To show the flexibility of our approach, we retarget facial and head motions from one person to another on in-the-wild videos that were established as a benchmark in [24]. In this setting we train the neural texture and renderer on our target subject and use these learned models at inference time, driven by the head motion reconstructed from a source sequence with a different actor. In Figure 9 we show that our approach is capable of retargeting with just transferring expressions (as in [24]) and also when mapping the global head orientation. The original results, provided from [24] do not include head stabilization. Nevertheless, this comparison shows that our approach (1x warp) is reproducing or even outperforming their image quality.

Operation Mode	L1 ↓	PSNR ↑	SSIM ↑
Ours (1x warp @ 30fps)	0.0361	23.19	0.8148
Ours (2x warp @ 60fps)	**0.0359**	**23.23**	**0.8162**

Table 2: ***high-fps* comparison** for our warping network using realistic operation settings on the *high-fps* dataset. 2x warp not only improves speed but even slightly improves the rendering quality.

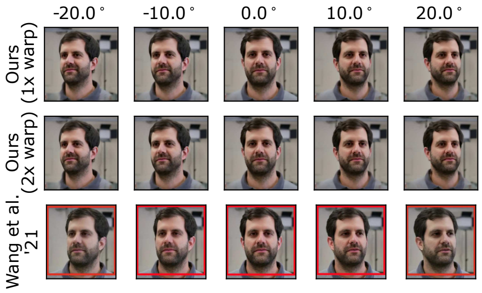

Figure 7: **Comparison against Wang et al. [30]**. While both models show similar levels of detail, ours is anchored in a 3D representation which gives us more fine-grained and independent control.

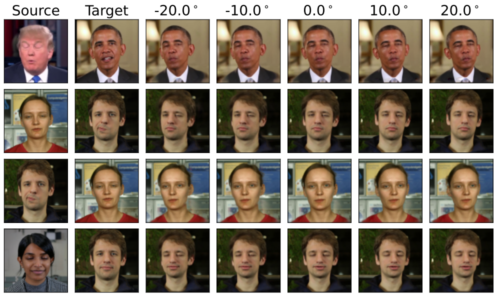

Figure 8: **Novel views while retargeting**. Our model is capable of generating realistic, view-dependent novels views (Section 5.4) while mimicking the source's facial expressions (Section 5.5). These examples are generated using Ours (1x warp) using the target actors from [24].

Furthermore, as we condition on a full 3D head model, we are able to generate novel views while performing the retargeting as shown in Figure 8. Since our approach only approximates the background for each scene, we use a static background in our predictions.

5.6. Ablation Study

Network Backbones. To show our method's generality to other backbones, we compare our UNet to the widely successful ResNet-backbone using residual blocks. As expected, warping is effective in reducing latency from 113 ms to 26 ms, but the ResNet does not outperform the UNet architectures. On the *beard* sequence, the ResNet backbone

Concat UV	Use Θ	Use MLP	SH Pose	SH Skips	ExWarp	Exp.	L1 ↓	PSNR ↑	SSIM ↑	Latency [ms] ↓	Rel. Latency [ms] ↓
✓							0.0276	25.6799	0.8987	12.80	-
✓	✓						0.0275	25.6994	0.8989	13.24	0.44
✓	✓	✓					0.0271	25.8378	0.9012	13.61	0.81
✓	✓	✓	✓				0.0260	26.1627	0.9038	13.79	0.99
✓	✓	✓	✓	✓			0.0244	26.5705	0.9105	14.40	1.60
✓	✓	✓	✓	✓	✓		0.0257	26.3326	**0.9108**	18.84	6.03
✓	✓	✓	✓	✓		✓	**0.0244**	**26.6562**	0.9107	14.62	1.82

Table 3: **Ablation study results** for our warping network using the sequential variant (1x warp), showing the relative improvement resulting from including each component on the *beard* dataset.

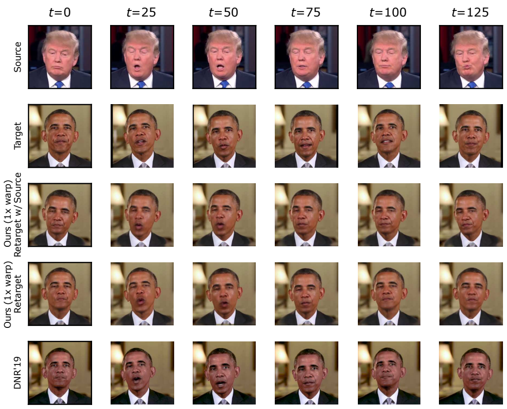

Figure 9: **Retargeting**, without and with transfer of the global head motion, including the comparison to [24].

operating using single warps gives a PSNR of 23.70 (UNet 26.66) and 23.18 PSNR using two warps (UNet 26.45).

Model Components. Adding the parameters and transformations used in *Our* full model one-by-one increases image quality (L1, PSNR, and SSIM) while only incurring a slight decrease in latency and fps (see relative latency and relative fps column, the relative latency compared to our full model). Table 3 presents the results for the sequential operation mode on the *beard* sequence over the following components:

- **Concat UV:** using the difference of the current and cached UV maps in $\mathbf{C}_3$.
- **Use θ:** concatenating the cached pose and camera extrinsics to $\mathbf{C}_3$.
- **Use MLP:** passing θ, $\mathbf{e}$, $\mathbf{p}$ through an MLP M before concatenation with $\mathbf{C}_3$.
- **SH Pose:** includes the spherical harmonics S^{obj} in θ.
- **SH Skips:** 'rotationally' encode $\mathbf{C}_4, \mathbf{C}_5$ using $\Delta\mathbf{S}^{obj}$.
- **ExWarp:** explicit warping from a reference by sampling the learned neural texture with $\mathbf{U}_{t+1}$ and concatenating it with $\mathbf{C}_3$.
- **Exp:** cache and concatenate the expression $\mathbf{e}$ with θ.

The explicit warping (*ExWarp*, second-last row) adds a large latency increase while not improving the quality metrics consistently. Hence, we favour the implicit warp in our full model. When applied in parallel on multiple GPUs, these reduced latencies translate directly to improved frame rate. The synchronization overhead in the parallel implementation is only 0.25ms/frame, which we measured by running the sequential model with the same threading and queue synchronization as for the parallel mode and taking their latency difference.

6. Limitations

Because our generator, G, and warping network, W, learn how to generate an image without an explicit rendering equation, we require a diverse set of training views to ensure that we can perform novel view synthesis and accurate warping at inference time. We can see in Figure 8 that the novel views break at extreme angles around the ears of the target subject as these are not seen during our training videos. The limiting factor is here the face reconstruction algorithm that becomes unreliable when large parts of the face are occluded. Eyes and a wide-open mouths can also pose a problem since they are represented as holes in the underlying FLAME model and therefore the direction of the eye gaze and tongue cannot be modeled. Furthermore, because the FLAME model only estimates 3D models for the head, the network struggles in cases where users have glasses or expressions that it cannot express. Improving image quality in these directions is largely orthogonal to our contributions towards low latency and high frame rate.

7. Conclusion

We introduced an implicit warping method that reduces the latency and, if parallel hardware is available, increases the frame rate of neural face rendering. We believe that such parallel execution to reduce latency and increase frame rate will gain importance with VR and AR emerging on the consumer market at scale, as our caching approach is compatible with the deeper neural network architectures required to meet the ever-increasing demand for output resolution. Thus, our work makes an important step towards end-to-end VR and 3D telepresence using view-dependent displays.

Acknowledgements This work was sponsored in part by the Natural Sciences and Engineering Research Council (NSERC), Huawei, and Compute Canada. We also want to thank Justus Thies for providing their dataset and baseline.

References

[1] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. 1999.

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[3] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 649–666, 2018.

[4] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3D imitative-contrastive learning. In *CVPR*, 2020.

[5] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020.

[6] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. volume 40, 2021.

[7] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *ACM TOG*, 2019.

[8] Jiahao Geng, Tianjia Shao, Youyi Zheng, Yanlin Weng, and Kun Zhou. Warp-guided GANs for single-photo facial animation. *ACM TOG*, 2018.

[9] Partha Ghosh, Pravir Singh Gupta, Roy Uziel, Anurag Ranjan, Michael J Black, and Timo Bolkart. Gif: Generative interpretable faces. In *2020 International Conference on 3D Vision (3DV)*, pages 868–878. IEEE, 2020.

[10] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8827, 2020.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2017.

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[13] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *arXiv preprint arXiv:2106.12423*, 2021.

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

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

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[18] Mateusz Malinowski, Grzegorz Swirszcz, Joao Carreira, and Viorica Patraucean. Sideways: Depth-parallel training of video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11834–11843, 2020.

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

[20] Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. paGAN: real-time avatars using dynamic textures. *ACM TOG*, 2018.

[21] Kyle Olszewski, Zimo Li, Chao Yang, Yi Zhou, Ronald Yu, Zeng Huang, Sitao Xiang, Shunsuke Saito, Pushmeet Kohli, and Hao Li. Realistic dynamic facial textures from a single image using GANs. In *ICCV*, 2017.

[22] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing Obama: learning lip sync from audio. *ACM TOG*, 2017.

[23] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt. StyleRig: Rigging StyleGAN for 3d control over portrait images. In *CVPR*, 2020.

[24] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[25] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM TOG*, 2015.

[26] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[27] JMP Van Waveren. The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 37–46, 2016.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[29] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popovic. Face transfer with multilinear models. *ACM TOG*, 2005.

[30] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10039–10049, 2021.

[31] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *arXiv preprint arXiv:2103.14024*, 2021.

[32] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, 2020.

[33] Qian Zhou, Georg Hagemann, Dylan Fafard, Ian Stavness, and Sidney Fels. An evaluation of depth and size perception on a spherical fish tank virtual reality display. *IEEE transactions on visualization and computer graphics*, 2019.

[34] Qian Zhou, Gregor Miller, Kai Wu, Ian Stavness, and Sidney Fels. Analysis and practical minimization of registration error in a spherical fish tank virtual reality system. volume 10114, pages 519–534, 03 2017.

[35] Qian Zhou, Kai Wu, Gregor Miller, Ian Stavness, and Sidney Fels. 3dps: An auto-calibrated three-dimensional perspective-corrected spherical display. In *2017 IEEE Virtual Reality (VR)*, pages 455–456. IEEE, 2017.