DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Audio-driven Neural Rendering of Portrait Videos

Audio-basiertes Neural Rendering von Portrait Videos

Author:	Wojciech Zielonka
Supervisor:	Matthias Nießner
Advisor:	Justus Thies
Submission Date:	February 15th 2021

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, February 15th 2021 Wojciech Zielonka

Acknowledgments

I would like to express special thanks of gratitude to my supervisor Matthias Nießner for the opportunity of doing my research in his unique Visual Computing Laboratory. Furthermore, I would like to thank my advisor Justus Thies sincerely. His expertise and knowledge laid the foundations without which this project would not be possible. Finally, I would like to thank my family for their support and love.

Abstract

This thesis proposes a novel way of synthesizing audio-driven portrait videos. We show that photo-realistic images can be rendered based on a small, fully connected neural network with the positional encoding of 3D face surface and additional audio-features extracted from an arbitrary English speech. The method is based on the intermediate geometry of 3DMMs. However, it is restricted neither by any face model in particular nor by its expression or identity space. The pipeline predicts both RGB color and 3D vertex displacement with respect to the mesh's neutral space for a given speech. Temporal stabilization for audio-feature vectors filtering provides smooth lip-audio synchronization. The rendered face is seamlessly embedded into a background using the autoencoder network with 2D dilated convolutions. Furthermore, the method generalizes well for an arbitrary speech from an unknown source actor on the condition that the English language is used. Finally, some state-of-the-art projects were selected for the method evaluation. Our method outperforms all of them in terms of image quality while maintaining low lip synchronization error.

Contents

1. Introduction

Generating photo-realistic images has always been the biggest challenge of computer graphics. Many sophisticated algorithms have been developed throughout the years. The most popular ones are based on the rendering equation introduced by James Kajiya [1]. However, the biggest problem with those approaches is the necessity of having accurate scene descriptions with geometry, lights, and materials manually crafted by artists. It creates a bottleneck in terms of time, realism, and resources. As an alternative, in recent years, a new solution has been developed. With the advent of neural networks, researchers have created compelling algorithms, which can generate photo-realistic images without any light transport equation[1] underneath. Generative Adversarial Neural Networks (GANs) by Goodfellow et al. [3] are an example of such a model, which in recent years evolved into more advanced high-resolution image generation [4]. Jun-Yan Zhu et al. [5] expand this concept to controllable image creation by conditioning networks on control parameters. Finally, those two areas have merged under a term known as Neural Rendering, which Tewari et al. [6] define as:

"Deep image or video generation approaches that enable explicit or implicit control of scene properties such as illumination, camera parameters, pose, geometry, appearance, and semantic structure."

The audio-driven generation of animated portrait videos of human faces is an excellent example of Neural Rendering. It requires capturing all intricacies of a face shape, facial expressions, skin reflectance, and the most critical, facial motion. It is a very challenging problem since facial motion and speech expose a strong correlation. However, they belong to two different domains. Additionally, any inconsistencies in the final video can be immediately spotted by a viewer. Humans are sensitive to lip movement and audio synchronization, which makes this problem even more difficult. However, the advent of neural networks has brought the interest of researchers in facial reenactment [7, 8, 9] who lately have made massive progress in many different applications from video dubbing [10] to generating and detecting fake news [11].

Many current methods [12, 13, 14, 15] show that it is possible to generate a digital face as a function of a 3D model and drive all non-rigid deformations in this domain. However, the challenge is first, how to obtain a detailed 3D shape model of a face and second, how to perform facial motion based on an arbitrary speech. For the first problem, a popular method is 3D morphable face models (3DMM) [16, 17, 18], for more details, Egger et al. [19] prepared

[1] "The light transport equation is the governing equation that describes the equilibrium distribution of radiance in a scene. It gives the total reflected radiance at a point on a surface in terms of emission from the surface, its BSDF, and the distribution of incident illumination arriving at the point." [2]

an excellent state-of-the-art survey about this topic. 3DMMs are widely used for image-based reconstruction using analysis-by-synthesis [17, 19] and more recently, together with deep learning methods, create a powerful toolset for this task. All of that produces a relatively easy to use framework, which can reconstruct a face template with expressions and identity vectors from a short RGB video in real-time [8]. Thies et al. [12] extend this concept by estimating 3DMMs expressions vector based on audio and producing photo-realistic images using 2-3 minutes videos of target actors for neural network training. However, this method requires an exact UV mapping for the texture domain since the rendering network is using neural textures [20].

Another approach for facial reenactment is based on deep encoder-decoder networks for image generation [9, 21, 22]. Those networks are capable of creating realistic images driven by audio. Compared to the methods mentioned earlier, the main advantage is that the intermediate 3D representation does not constrain the solution space. Therefore, it is universal and can be applied to every face. However, at the same time, without a 3D model, it is impossible to apply an affine transformation to a digital avatar, which is a crucial aspect in some applications, for instance, video games.

Addressing weaknesses of the previously mentioned approaches, we propose a novel method for generating photo-realistic videos of talking heads driven by audio. Based on the recent research in neural renderers [23, 24, 25] we show that a simple fully connected network with positional encoding [23] and additional input as audio features [9, 12] can generate photo-realistic talking heads together with the estimation of 3D mesh function and its non-rigid deformation. In summary, this work proposes a new approach for generating audio-driven videos with the following:

- The digital avatar can be trained using 2-3 minutes portrait videos.
- The generated video preserves the talking style of a target actor.
- Together with photo-realistic images, the network produces a 3D resolution-independent template mesh of a given actor.
- The talking head is driven purely by audio input.
- The method is independent of used 3DMMs for data generation.
- Simple and compact (3 million parameters) neural rendering method for audio-driven video generation.

2. Related Work

Photo-realistic face rendering is a recurring topic in many research areas. Extending it with speech-driven animation presents a very challenging problem, which has many exciting approaches. They can be grouped into three main categories: speech-based, text-based, video-based, or performance-based [14]. This chapter elaborates those methods emphasizing, first, neural rendering, as it is the core method of this work, then focusing on realistic face animation using 3DMMs. Face synthesis without intermediate geometry is explained next. Finally, it covers deep generative adversarial networks, which gained much attention since Karras et al. have presented StyleGAN [26, 27].

2.1. Physically Based Rendering

Before defining Neural Rendering, it is crucial to explain Physically Based Rendering (PBR) first. PBR is a simulation of light transport process [2], a physical model that statistically describes photons' interactions with objects in the scene. A classic formulation of this problem is the rendering equation [1]

$$L_o(\mathbf{p}, \omega_o, \lambda, t) = L_e(\mathbf{p}, \omega_o, \lambda, t) + L_r(\mathbf{p}, \omega_o, \lambda, t), \tag{2.1}$$

where L_o represents exitant radiance from a point $\mathbf{p}$ in the direction ω_o. L_e describes emitted radiance and L_r is a fraction of incident radiance scattered by a surface material [2]

$$L_r(\mathbf{p}, \omega_o, \lambda, t) = \int_\Omega f_r(\mathbf{p}, \omega_i, \omega_o, \lambda, t) L_i(\mathbf{p}, \omega_i, \lambda, t) |cos\theta_i| \, d\omega_i. \tag{2.2}$$

Unfortunately, because of the infinitely recursive nature of the equation, the integral 2.2 cannot be solved in the closed-form. Therefore, an approximation has to be used. The most popular ones are based on Monte Carlo Integration [28, 29]. Additionally, the quality of rendering heavily depends on $f_r(\mathbf{p}, \omega_i, \omega_o, \lambda, t)$ which is the bidirectional reflectance distribution function (BRDF) [2]. This function describes the material model, meaning how light interacts with a given surface. Many companies, like Disney [30], Electronic Arts [31], or Epic Games [32] invested much time creating their own BRDFs. However, in the end, to get a photo-realistic rendering, an artist has to manually craft a scene with geometry, material parameters, and lighting setup. The whole process is incredibly complicated and time-consuming. Many researchers try to simplify it by developing tools: automatic BRDF editing and creation [33], or appearance-based search for material suggestions [34] which help artists to save time.

To summarize, PBR requires an accurate scene description. However, simultaneously giving the ability to manipulate all scene's components: camera, geometry, materials, lighting. This scene control is still far out of reach for neural rendering methods.

2.2. Neural Rendering

Neural rendering is a method that tries to provide a middle point between deep generative models (Sections 2.5, 2.4) and classic rendering methods with high-quality controllable image synthesis (Section 2.1). This approach, based on certain scene elements: lighting, camera, textures, generates a neural representation from which novel images can be synthesized. It is a powerful concept, however, for now, restricted to small scenes only.

Generalization of neural rendering is a challenging and active research area. Current methods tackle only some specific sub-problems [6]. One of them is novel view synthesis [23, 35, 36, 37, 38]. Recent work by Rahaman et al. [39] opened an exciting chapter in neural image synthesis, as they showed that mapping input to higher dimensional space could significantly improve the prediction of high-frequency variation in data. Based on that discovery, Mildenhall et al. [23] introduced positional encoding for image synthesis. Using a simple fully-connected deep network and volume rendering, they created a neural radiance field from which photo-realistic novel views can be synthesized. Gafni et al. [36] extended this concept to dynamic neural radiance fields for faces. They combined NeRF [23] with 3DMMs, resulting in dynamic conditioning of facial expressions over neural volume. Srinivasan et al. [35] presented an improved version of NeRF, which they called Neural Reflectance and Visibility Fields (NeRV). They included the simulation of light transport, therefore, allowing image synthesis under novel illumination conditions. It is a classic inverse rendering problem [40, 41], where except geometry, light, and BRDF parameters have to be estimated as well.

Deferred Neural Rendering introduced by Thies at el. [20] redefines usage of classical computer graphics textures [42]. Neural textures describe learned features for a given point. Therefore, the network requires coarse geometry with a texture parametrization for later sampling using a computer graphics rasterizer. The pipeline allows rendering photo-realistic novel images as well as scene editing.

The neural talking heads models is another example of a neural rendering application. There are two major approaches, one is based on intermediate geometry [12, 36, 13, 14, 43, 18, 10], usually 3DMMs, and the other is direct synthesis without any intermediate layer, or proxy [22, 21, 9, 44, 45]. Sections 2.3 and 2.4 give an overview on those two solutions.

2.3. Parametric Face Synthesis

Speech-driven face synthesis is a topic with many potential applications. From video dubbing to teleconferencing or photo-realistic virtual avatars [7]. Working on a raw pixel stream is difficult due to the high-dimensional nature of images. Therefore, one approach to solve this problem is using an intermediate proxy; usually, 3DMMs [19].

Face representation as a parametric model optimized by analysis-by-synthesis was an essential contribution by Blanz and Vetter in 1999 [17]. Figure 2.1 shows introduced in their paper a holistic framework that reconstructs a 3D face from 2D images.

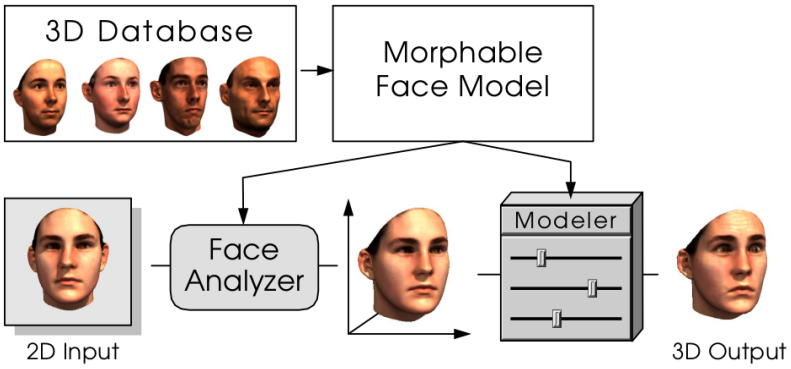


Figure 2.1.: 3DMMs analysis-by-synthesis framework by Blanz and Vetter. Image from [17].

The parametric face model is created by performing the principal component analysis (PCA) on the face database to extract eigenvectors and create eigen basis [46, 47] where the optimization process takes place. Shapes in the database have to be in dense, full correspondence, which is crucial for PCA. Yenamandra et al. [48] extend this concept by creating i3DMM, which except the frontal part of a face, additionally, models hair and full head. The model's parameters are optimized using optical flow, minimizing the photo-consistency error of the projected 3D model into the 2D image and the target 2D image (differentiable rendering). 3DMMs create a constrained space spanned by eigenvectors since matching a 3D surface to a given image produces many non-face-like surfaces [17].

Before focusing on speech-driven 3DMMs, methods with a different proxy representation have to be introduced. Voice Puppetry by Brand [43] uses tracked facial landmarks to the piecewise approximate underlying manifold. This coarse representation is predicted using a Hidden Markov Model (HMM). The transition probabilities for each pose are sampled from the HMM. During inference, the Viterbi algorithm [49] is used to identify the most likely state sequence given a signal [43]. The iFace system developed by Hong et al. [50] based on labeled facial deformations and mel-frequency cepstral coefficient (MFCC) with motion units can produce audio-driven images. Chen et al. [51] proposed a cascade GAN based on facial landmarks predicted from audio and a novel loss function with an attention mechanism generating temporal stable images driven by speech.

The active appearance model (AAM) introduced by Cootes et al. [52] is yet another way of reconstructing faces. It consists of statistical models of shape variation and gray-level appearance created by using grayscale images with 122 landmark points. One of AAMs' advantages is the explicit correlation between shape and texture, which is a constraint during statistical analysis. In the same spirit, Zhou et al. [53] proposed joint texture and shape in-the-wild auto-encoder for optimizing 3DMMs.

The advent of deep neural networks allowed researchers to rediscover 3DMMs and improve

on them [19]. Many of the solutions, first, perform speech-driven facial animation in the intermediate representation space, and based on that result, the final image is rendered. Thies et al. [12], using an only audio signal, predicts the coefficients of a parametric face model, which later is used as the input to the rendering network. For this purpose, audio-features are extracted using DeepSpeech [54] and projected from audio-feature space to 3DMMs expression space [17] using Audio2Expression neural network. For face tracking and data generation, they used Face2Face [8] framework. One of the downsides of this solution is the need to train the pipeline for every target actor. Pham et al. [15] use the FaceWarehouse database [55] to represent a blendshape face model. In their work, an LSTM-RNN model learns how to project speech to facial parameters of the model. FLAME [18] is an example of a new robust 3DMM that addresses the limitations of previous 3DMMs by introducing a more accurate and expressive model compared to [16, 17] and additionally, modeling head pose and eyeball rotation. Based on that work, Cudeiro et al. introduced VOCA [14]. Similarly to Thies et al. [12] they used DeepSpeech [54] to extract robust audio-features. FLAME [18] is used to obtain underlying face models from 3D scans creating VOCASET subjects. The network is then fed with speech features and FLAME's template to predict vertex displacement w.r.t the neutral template's state. Similar work was done by Karras et al. [13]. However, in contrast to the aforementioned methods, it does not use DeepSpeech. Instead, the audio-feature prediction is incorporated into the pipeline and jointly optimized to learn the mapping from audio to the 3D vertex coordinates of a face model in an end-to-end fashion. An important aspect of this work is an additional input vector that controls the target's emotional state, thus greatly influencing the speaking style. Tzirakis et al. [56] developed a speech blendshapes model designed for better audio-driven animation of a 3D face mesh.

Additionally, they presented Deep Canonical Attentional Warping (DCAW), which projects an audio signal to their model's expression parameters space. DCAW is used to estimate the alignment path between the audio signal and Lip Reading Words (LRW). This allows them to generalize well to every speaker. Methods with intermediate representation have one disadvantage. They are usually constrained by the solution space of a given 3DMM. Therefore, researchers have been investigating direct face synthesis, without geometry, using only a raw stream of pixels and an audio signal.

2.4. Explicit Face Synthesis

Section 2.3 elaborates different approaches to generate audio-driven portrait videos using the underlying geometry. Many of those solutions have to be retrained for each target actor. It is time-consuming and does not generalize well. On the other hand, direct rendering is neither constrained by 3DMMs nor is it target specific.

Deep generative adversarial networks (GANs) are very popular for synthesizing novel images. Prajwal et al. [21] introduced LipGAN, which generates realistic talking heads from an audio signal. Their main goal was an automatic lip-synchronized video rendering. Using text-to-speech synthesis, the framework can generalize well over many actors and languages. In the consequent work, Prajwal et al. [9] improved results using a pre-trained Lip-Sync

Expert as a discriminator to detect if audio and video are synchronized. Figure 2.2 presents Wav2Lip pipeline. It uses two encoders for audio and face to generate embedding vectors $\in \mathbb{R}^{512}$ which are later concatenated and fed to the face decoder. The audio encoder uses MFCC as input, whereas, face encoder operates on half-masked segments concatenated with random reference segments. It helps to synthesis a new face image with the correct pose.

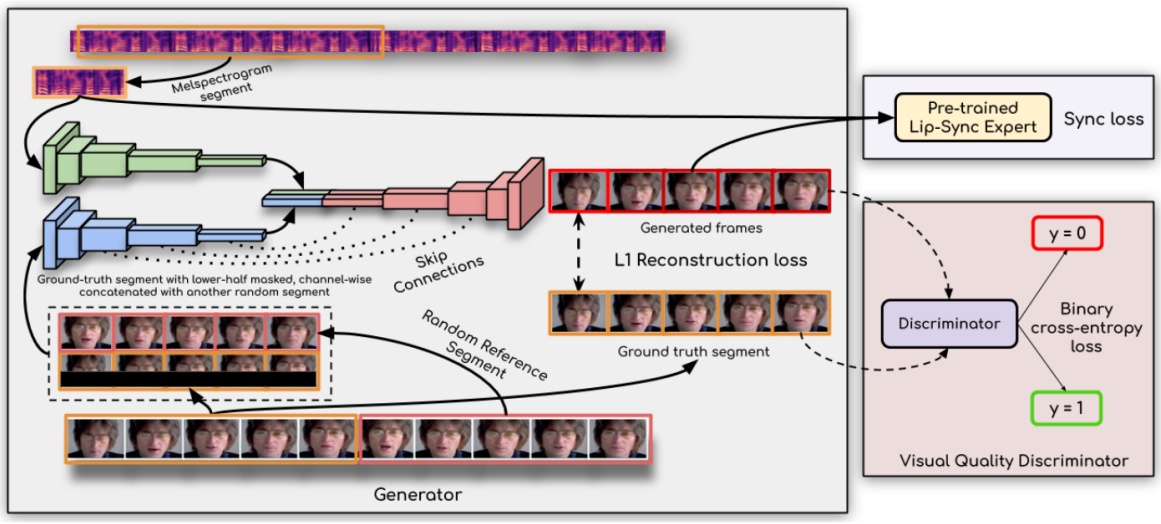


Figure 2.2.: Wav2Lip architecture overview. The generator consists of two encoder networks and one decoder, which uses concatenated embedding vectors from the encoders as the input. The network has two discriminator networks. One is based on the pre-trained lip-synchronization network, and the other measures image quality based on the L1 loss. All of that gives a framework to produce realistic images for arbitrary speech. Image from [9].

Chung et al. [57] presented the Speech2Vid pipeline, which using audio MFCC, renders novel images. The result, however, needs additional post-processing using the Deblurring module. The architecture is divided into three main parts. Audio and identity encoders generate a joint encoding, which later is used by the image decoder to render talking faces. In contrast to Speech2Vid, Suwajanakorn et al. [44] developed a network for synthesizing one actor only. They used training data constituted of 17h video of Barack Obama. The rendering process is divided into two steps. First, using a recurrent neural network, sparse shape coefficients are learned based on MFCC. Next, a synthesis of Obama's high detailed textures is performed. Vougioukas et al. [58] use temporal GANs to produce coherent results. The RNN-based generator is fed with still images and audio. The audio frame is encoded with a 256 elements vector, which later goes through RNN layer to capture temporal changes. In the next section, StyleGAN [26, 27] is elaborated in more details, as it got a lot of attention recently.

2.5. Deep Generative Adversarial Networks

Karras et al. [26, 27] have shown that very deep GANs are able to generate random, indistinguishable from reality, images. It encouraged researchers to investigate the ways to control StyleGAN [26, 27] by exploring its latent space.

Tewari et al. [59] suggested using 3DMMs as a way to control pre-trained StyleGAN. RigNet [59] projects latent code $\mathbf{w}$ and 3DMM's parameters vector $\mathbf{p}$ (albedo, facial expression, scene illumination, facial shape, pose) to extended latent space. This new latent code is used as input to StyleGAN to produce a modified face image. RigNet is trained in a way that makes it possible to inject a new subset of parameters, for instance, a different expression vector of 3DMM, and in this way control the resulting image. This research gives hope for pipelines, where 3DMMs are controlled by audio, and indirectly using RigNet, photo-realistic images with teeth, hair, eyes are generated by StyleGAN. Meissen [60] proposed a direct way of controlling StyleGAN2 image generation by using audio features [12] only. Similarly to Tewari et al. [59] he computes extended latent code. However, he does not use 3DMMs as a middle layer for this purpose. Instead, AudioStyleNet is based on audio features, and a still image predicts the extended latent code.

3. Method

Generation of photo-realistic speech dependent portrait videos is the main goal of this work. Several components were used to achieve it, creating a holistic pipeline, which can automatically produce videos of talking heads. This chapter gives an overview of the framework, elaborating in-depth on each of the components. Throughout Section 3.3 till Section 3.6 it covers the design of the neural networks. Finally, in the last Section 3.7 loss functions are elaborated.

3.1. Overview

The framework has four main components, audio-features extraction (Section 3.3), projection of extracted embedding to a different, lower dimensional space (Section 3.4). The deformation and color networks (Section 3.5) which are the core of the pipeline. Finally, the composition network (Section 3.6) which is trained separately, therefore, it is not included in the pipeline (Figure 3.1).

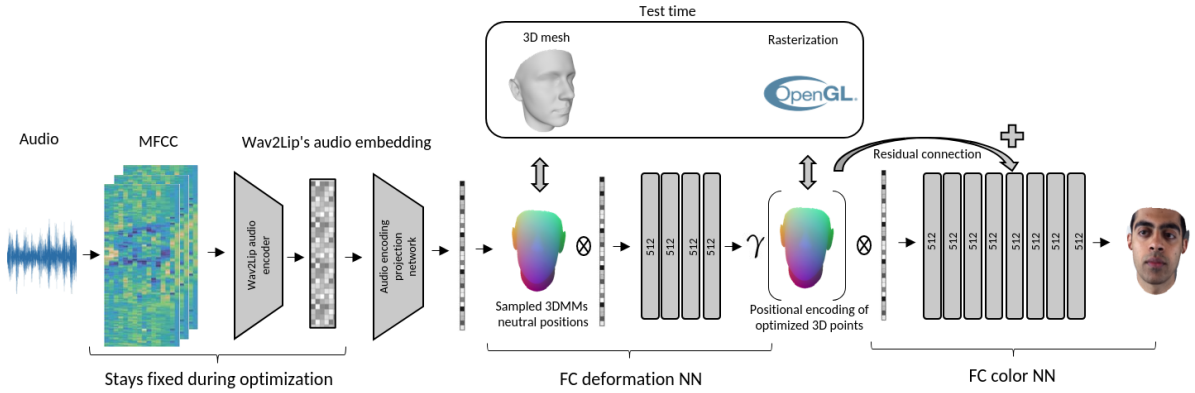


Figure 3.1.: Main pipeline architecture overview. The network accepts an audio signal as input. MFCC features are extracted and audio embedding $\in \mathbb{R}^{512}$ is generated by Wav2Lip's [9] audio encoder. The output is projected to $\mathbb{R}^{64}$ resulting in vector $\mathbf{b}$ which is concatenated with 3D neutral face points. The deformation network approximates 3DMMs expression and identity functions producing an optimized 3D shape. This shape is encoded using positional encoding $\gamma(\cdot)$ from [23] and again concatenated with $\mathbf{b}$. Finally, the color network outputs photo-realistic images driven purely by audio.

The model formulation is as follows; given speech A which is divided into a sequence $A_1, A_2, ..., A_k$. The pipeline has to generate sequence of frames $F_1, F_2, ..., F_k$ for a given target actor. It means that for every actor, a separate network has to be trained. The network's input is an audio signal, later converted to mel-frequency cepstral coefficients (MFCC), together with the sampled 3D position of 3DMMs. Parametric face model is obtained using state-of-the-art software Face2Face [8]. However, the pipeline is not restricted to any 3DMMs solution.

MFCC is fed to Wav2Lip [9] which produces audio embedding $\mathbf{a} \in \mathbb{R}^{512}$. The latent space of Wav2Lip is highly non-linear and entangled in Wav2Lip's decoder. The audio-features live in $\mathbb{R}^{512}$, therefore, to accelerate training and make it usable for the next part of the pipeline, the projection network approximates [61] the transformation of vector $\mathbf{a}$ to $\mathbb{R}^{64}$ space producing $\mathbf{b} \in \mathbb{R}^{64}$. More information can be found in Section 3.4. The deformation network has to approximate expression and identity functions from 3DMMs, producing optimized shape for a given frame based on the audio signal. The network's input is sampled as 3D points in the face neutral state concatenated with vector $\mathbf{b}$. During the test time, a 3D mesh is used instead. After predicting the vertex displacement, the mesh is rasterized using OpenGL. Next, the output from the deformation network is passed to the color network. Every 3D point undergoes positional encoding [23] and, similarly to the previous network, is concatenated with an audio embedding. The input is additionally added to the 5th hidden layer as a residual connection [62]. The final output is a photo-realistic image driven by speech. Both of those networks are 512 fully-connected multilayer perceptrons [61]. The following sections focus on each part of the pipeline in depth.

3.2. Neural Voice Puppetry: Audio Embedding

Audio-features are one of the most critical aspects of this work. They make the speech-driven approach possible. At the beginning of the project, audio-feature vectors from the Neural Voice Puppetry (NVP) [12] were used. Figure 3.3 presents the pipeline which generates audio embedding for a given audio segment. It was a natural choice because the NVP project operates in the 3DMMs linear space. Therefore, its audio-feature vectors are used to approximate the expression parameters of the face model.

The basic idea is first to extract audio-features vectors using DeepSpeech [54]. Every audio segment of the length $20ms$ a vector of features is predicted using the recurrent neural network architecture. Therefore, for each frame, the network produces an embedding of size 16×29. Next, the DeepSpeech result is fed to the Audio2ExpressionNet [12] network to produce the audio-expression vector. A vector in the unfiltered feature space is produced using a series of four convolutions and three fully connected layers. The unfiltered space is temporally unstable. Thus, the attention mechanism is incorporated to project the vector into a stable latent space. The result is a smooth and coherent audio-expression vector $\in \mathbb{R}^{32}$. The same attention mechanism was used in our project (Section 3.4).

During the project's development, several techniques were used to improve the result. NVP audio-features alone were not sufficient enough to capture all facial expressions. Therefore,

the auto-decoder network was introduced. Park et al. [63] presented the idea of the auto-decoder network in their DeepSDF project. The concept is to have a set of latent codes $\mathbf{z}$ which is conditioned on the output of the network (Figure 3.2). In our case each frame in a video had it's own latent code $\in \mathbb{R}^{32}$ which was later concatenated together with NVP audio-feature vector resulting in vector $\in \mathbb{R}^{64}$. The latent codes were optimized during the training to help capture all intricacies of each frame.

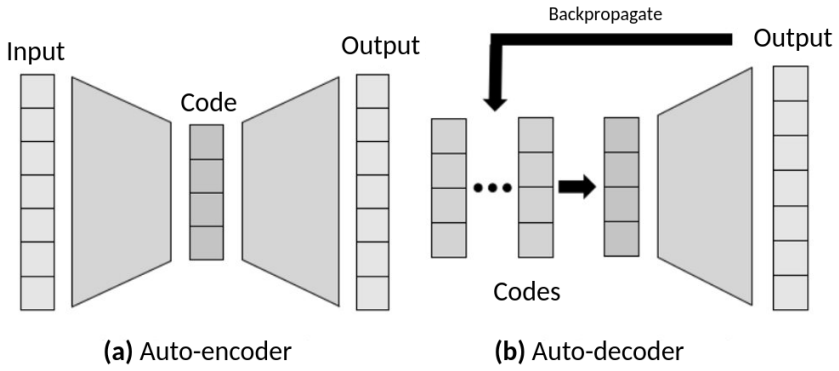


Figure 3.2.: DeepSDF auto-decoder architecture. Auto-decoder's codes are latent vectors which are sampled from Gaussian distribution. Because they are conditioned on the output, the optimization process shapes them during the learning process to represent a desired latent space. Image adapted from [63].

Unfortunately, because of the architecture of the renderer, the NVP audio embedding with the additional latent code did not work as expected. The results were characterized by wrong expressiveness of the facial motion. Despite many experiments and tries to improve the performance, this solution failed. Please refer to the ablation study section, in particular subsection 4.4.1, to see the comparison between usage of Wav2Lip and NVP audio-feature vectors.

3.3. Wav2Lip: Audio Embedding

As mentioned previously, the audio-feature vector from NVP did not work correctly with our network's architecture. After investigating alternatives, the best results were achieved using Wav2Lip [9] audio embedding vectors.

Prajwal et al. [9] used a GAN architecture with a pre-trained network based on Chung et al. [64] as the discriminator of audio-actor synchronization. Compared to their previous work LipGAN [21], they discovered that a discriminator trained using only a reconstruction loss very poorly detects off-sync audio-lip with accuracy around 56% [21]. Building on this fact, they proposed a new synchronization loss based on a pre-trained network specializing in this one task only. They call it Lip-sync Expert, and it is a modified version of SyncNet [64]. SyncNet is an encoder-decoder network with a stack of 2D-convolutions. It operates on a time window with length D. The inputs are grayscale images and the audio signal both

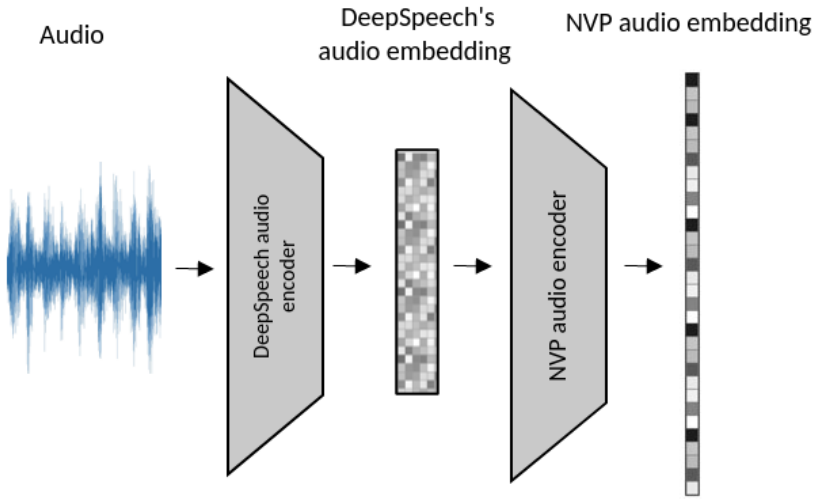


Figure 3.3.: Neural Voice Puppetry audio-feature vector generation. The input audio is first transformed using DeepSpeech to audio-features. Next, operating on those features NVP audio embedding is predicted in a temporarily stable space. NVP uses those embeddings to predict expression basis of the 3DMM.

divided into segments.

Wav2Lip does the following changes in contrast to SyncNet. Instead of using grayscale images, color images are fed to the network. The model architecture was made much deeper with residual skip connections [62]. Finally, a different loss function is used. They compute a cosine between the ReLU-activation function [65] of speech and video embedding v, s to get a probability which describes input audio-video synchronization between [0, 1] [9]

$$P_{sync} = \frac{v \cdot s}{max(\|v\|_2 \cdot \|s\|_2, \epsilon)}, \tag{3.1}$$

where ϵ is a small number to avoid division by zero. The discriminator trained in this way achieves 91% accuracy at detecting off-sync videos [9]. This process hugely improves the quality of generated images since using a Lip-sync Expert forces the generator to create precise lip shapes [9]. Additionally, the network is trained in a frame-by-frame fashion. This allows extracting audio or video embedding for a single frame easily. The network is trained using the publicly available LRS2-BBC [66] dataset, which contains thousands of natural sentences from British TV.

Table 3.1 represents Wav2Lip audio encoder's 2D-convolutions used in the pipeline. The first four layers have additional residual connections. Encoder-decoder network is used as GAN's generator, which together with Lip-sync Expert as a discriminator gives state-of-the-art architecture for lip-audio synchronization. Figure 2.2 gives a holistic overview of the whole pipeline.

Type	Kernel	Stride	Output	Activation
2D convolution	3	1	$32 \times T$	ReLU
2D convolution	3	(3, 1)	$64 \times T$	ReLU
2D convolution	3	3	$128 \times T$	ReLU
2D convolution	3	(3, 2)	$256 \times T$	ReLU
2D convolution	3	1	$512 \times T$	ReLU
2D convolution	3	1	$512 \times T$	ReLU

Table 3.1.: Detailed list of Wav2Lip audio encoder's 2D-convolutions.

To summarize, Wav2Lip is an excellent speech-audio synchronization tool. The latent audio space contains information that successfully can be digested by our pipeline producing accurate facial motion driven by speech. During the optimization of our pipeline the pre-trained Wav2Lip was used and it stays fixed.

3.4. Projection Network

A projection network is a module in the pipeline responsible for approximating a transformation from Wav2Lip latent space to the one defined by the current actor. During experiments, first, the original audio-features produced by Wav2Lip were used. Unfortunately, the network could not easily converge due to the fact that those latent codes come from highly non-linear space discovered during Wav2Lip optimization process. A small 1D-convolutions network was introduced to alleviate this problem, alongside with an attenuation module to maintain temporal coherence for smooth video generation.

The whole projection stage is divided into two neural networks jointly optimized. The first one is a transformation approximator, and the second one is an attention module.

Type	Kernel	Stride	Output	Activation
1D convolution	1	1	$256 \times T$	Leaky ReLU
1D convolution	1	1	$256 \times T$	Leaky ReLU
1D convolution	1	1	$128 \times T$	Leaky ReLU
1D convolution	1	1	$64 \times T$	Leaky ReLU

Table 3.2.: Detailed list of projection network's 1D-convolutions.

Table 3.2 presents a detailed list of the first networks' convolutions. It is a simple series of 1D-convolutions with Leaky ReLU activation functions. The main purpose is to find an approximator g of a projection function $f : \mathbb{R}^{T\times 512} \rightarrow \mathbb{R}^{T\times 64}$, where T is number of frames in a window. The input to the network is matrix $\mathbf{A} \in \mathbb{R}^{T\times 512}$. However, using directly function g makes the results unstable in the temporal domain. The adapted version of SyncNet in Wav2Lip produces temporally stable results, but function g distorts the coherence. To fix this problem an attention mechanism [67] was introduced inspired from [12].

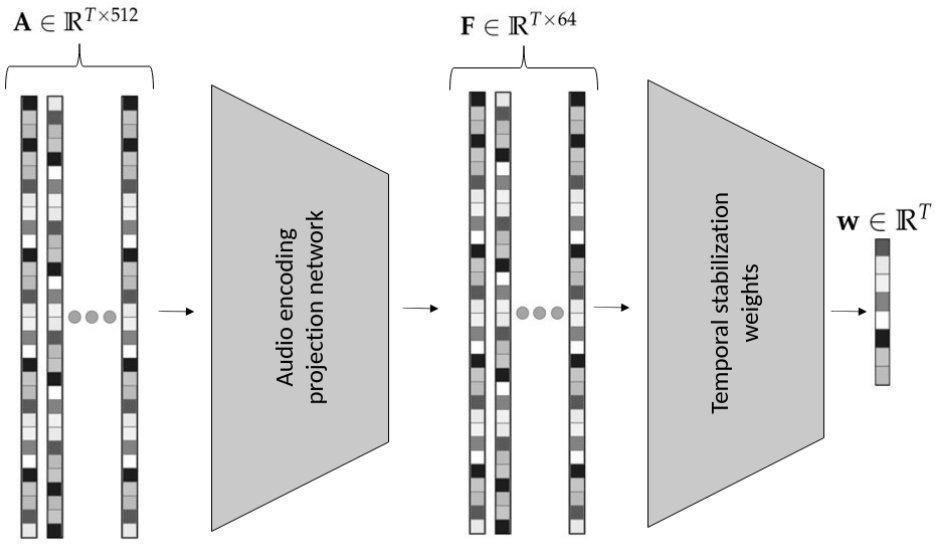


Figure 3.4.: Attention mechanism for audio-feature projection to a lower dimensional space $\mathbb{R}^{64}$. First the matrix $\mathbf{A}$ is projected into lower dimensional space in $\mathbb{R}^{T \times 64}$. Finally, a set of weights $\mathbf{w}$ is calculated for linear interpolation of the final vector $\mathbf{b} = \mathbf{Fw}$. Image adapted from [12].

The attention mechanism was introduced by Zhang et al. [67] to model long-range dependency for image generation problems. The main idea is to estimate the extent of attention a model has to put to a given location. Thies et al. [12] use a small neural network (Table 3.4) to find weights in the range between [0, 1] assigned to each frame in a window. They use a series of 1D-convolutions and softmax function $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$ to normalize the network's output such as the sum of weights is one by using

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{3.2}$$

The network predicts the vector of weights $\mathbf{w} \in \mathbb{R}^T$ (Figure 3.4), which later is used to calculate the final audio-features vector $\mathbf{b} = \mathbf{Fw}$, where $\mathbf{F} \in \mathbb{R}^{T \times 64}$ is a matrix of audio-features and $\mathbf{b} \in \mathbb{R}^{64}$. In other words, for a given time segment, it estimates how much attention a network has to put for each of the constituting frames.

During experiments, the projection network exhibited substantial overfitting if it was trained for one actor only. To avoid it, two techniques were used. The first one is noise injection as a way for parameters regularization [68]. Before passing the predicted audio-features to the next stages of the pipeline, the vector is normalized and summed with a random vector sampled from $N \sim (0, 1) * 0.2$. This process helps to enlarge the learning capacity of the color and deformation networks. The second is sequential training of the pipeline for at least two actors at the same time. The current implementation for each of the two actors initializes one

instance of the network, and each actor's loss is optimized w.r.t the instance's parameters separately.

Type	Kernel	Stride	Output	Activation
1D convolution	3	1	$16 \times T$	Leaky ReLU
1D convolution	3	1	$8 \times T$	Leaky ReLU
1D convolution	3	1	$4 \times T$	Leaky ReLU
1D convolution	3	1	$2 \times T$	Leaky ReLU
1D convolution	3	1	$1 \times T$	Leaky ReLU

Table 3.3.: Detailed list of attention network's 1D-convolutions.

3.5. Deformation and Color Networks

Deformation and color networks are the core part of the whole pipeline. For a given actor and scene they serve as audio signal based approximators of 3DMMs function and image function. Both networks are deep feedforward networks [69] which define an approximator $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta})$ with rectified linear units $g(z) = max(0, z)$ as the activation unit.

3.5.1. Deformation Network

The deformation network is responsible for predicting the displacement of 3D points with respect to their neutral position. It has four fully connected hidden layers, each with 512 nodes. The network accepts a 3D point as input which is concatenated with the previously described audio-features vector giving vector $\mathbf{d} \in \mathbb{R}^{(3+64)}$. Basel's 3DMM [16] used in Face2Face [8] (Section 4.1.1) is defined in a linear space spanned by eigenvectors. Thus, the network has to learn a linear transformation of a neutral 3D point to its non-rigid displacement in a space spanned by identity, shape, and expression basis of 3DMMs. The color network is a mapping $c : \mathbb{R}^{160} \rightarrow \mathbb{R}^3$ which based on a 3D position has to predict an RGB value of a given pixel. However, a 3DMMs' triangle mesh is just a piece-wise linear approximation of a face surface. Therefore, to estimate a 3D point for every pixel that covers the face mesh, interpolation must be performed. Rasterization process interpolates vertices of a triangle $T = \{a, b, c\} \in \mathbb{R}^3$ to a point which covers a given pixel (Figure 3.5) using barycentric coordinates [70]. Rasterizer samples a surface for each pixel, creating input to the networks. OpenGL was used in this project as a framework to generate the necessary data.

3.5.2. Color Network

Color network is the last part of the pipeline. It is responsible for generating photo-realistic images based on 3D positions. It has eight fully connected hidden layers, each with 512 nodes. Additionally, the fifth layer has a residual connection [62] to the input layer. As previously mentioned, neural networks are thought of as general function approximators [61]. However,

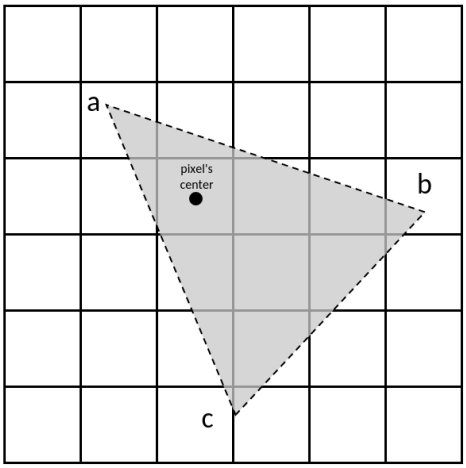


Figure 3.5.: Rasterization of the triangle $\{a, b, c\}$. For each pixel that constitutes a given triangle, the vertices' properties like color or position are interpolated using barycentric coordinates.

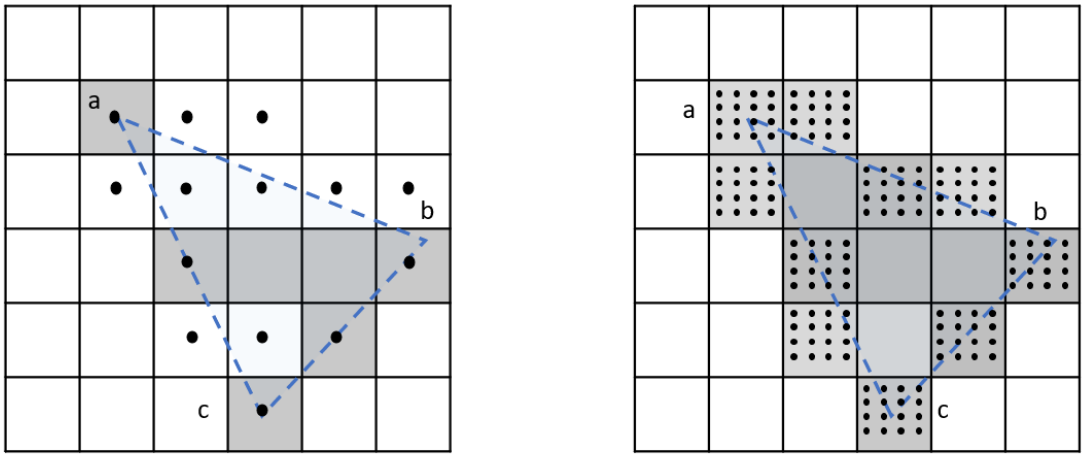
recent research shows that they poorly represent high-frequency parts of color and geometry functions [23].

Image rendering is the process of sampling a three-dimensional scene [70]. A continuous image function has to be represented as a discrete array of pixels. Unfortunately, rasterizing is inherently connected with aliasing. It is stated that for proper rasterization, the function sampling frequency has to be more than twice the maximum frequency of the sampled signal [70]. This limit is called Nyquist rate [71]. However, the image function is not band-limited, exposing infinitely high frequencies [72]. For instance, a 2D step function representing a line exhibits frequencies that cannot be sampled (Figure 3.6). The solution to avoid it would be a continuous computer and monitor, which is obviously not possible. The high-frequency parts can be band-limited by convolving the signal with a low-pass filter. Unfortunately, usually, it means blurring the image and losing quality. Photo-realistic images show many places with sudden frequency jumps that have to be captured by neural networks. Recent studies prove that a simple trick can help neural networks to learn high-frequency variations in data and continuously approximate the image function.

Rahaman et al. [39] show that high-frequencies are poorly represented by deep neural networks. In the image generation context it means that a neural approximator cannot well represent hard shadows, edges, specularities, etc. Therefore, resulting in a low quality image. Rahaman et al. [39] suggest to use a function which exhibits high-frequencies and map input to higher dimensional space [73].

Building on these findings, Mildenhall et al. [23] proposed positional encoding. They represent an image function F by $F = F^{'} \circ \gamma$, where $F^{'}$ is a regular MLP and $\gamma : \mathbb{R} \rightarrow \mathbb{R}^{2L}$ is a positional encoding kernel [23]

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), ..., \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)). \tag{3.3}$$



(a) Taking one sample per pixel is producing vast aliasing of the {a, b, c} triangle

(b) Increasing number of samples per pixel reduces aliasing (16x)

Figure 3.6.: Rasterization of 2D line function's infinitely high frequencies. The raster space is a discrete set of pixels. Therefore, the reconstruction of the image function is affected by aliasing exposing artifacts like jagged edges. Aliasing can be alleviated by increasing resolution or taking more samples per pixel.

The embedding is not optimized during the learning process. Every element of the input vector $\mathbf{p} \in \mathbb{R}^3$ is encoded using $\gamma(\cdot)$ before being passed to the color network. The number of frequency bands is $L = 16$. In a consequent work, Tancik et al. [25] presented Fourier Features as a kernel for mapping input vectors to higher-dimensional space. Exploring this aspect in the context of audio-driven images is left for future work.

3.6. Composition Network

The final part of the network, which is not jointly optimized with the rest of the pipeline, is a smooth "inpainter" of the generated image and background. For each source frame, meaning the original frame from the video which will be modified, the network alters it by replacing the mouth region with the one generated based on the input audio (Sections 3.5.1, 3.5.2). Figure 3.8 presents the architecture. First, the source frame is eroded, and the whole lower part of the face is removed. Next, the eroded background is summed with the rendered mask to create the input image. In this way, the network is forced to learn how the mouth composites with the rest of the face and background for a particular facial motion driven by audio. The main idea is that the network learns how to fill the seam in a meaningful way to produce a smooth and coherent frame. The network uses the Pix2Pix [74] framework that is modified based on the Neural Voice Puppetry idea [12]. Meaning, the strided convolutions in U-Net are replaced with dilated convolutions (Figure 3.7). Dilated convolution significantly increase the receptive field of a CNN without loss of resolution. Therefore, the output pixel is

generated based on all important information from the input signal [75], which helps produce photo-realistic images.

Additionally, transposed convolutions are replaced with standard ones since the encoder does not downsample the image. All kernels are 3x3 size and are followed by the leaky ReLU activation function. The network has approximately 2.35 million parameters with 32 features per layer, which is less than the pipeline to generate an audio-driven image (approximately 3 million parameters).

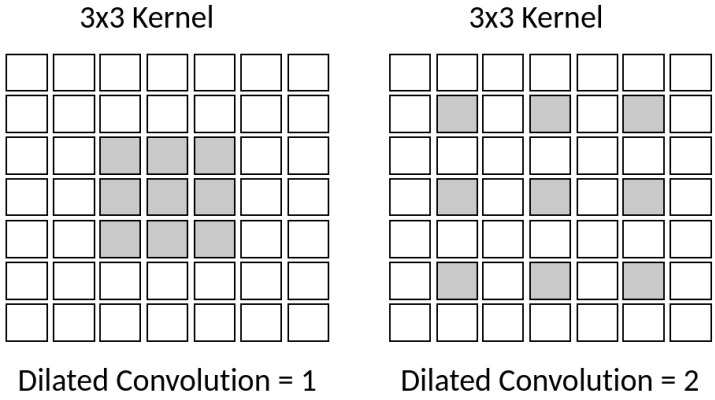


Figure 3.7.: Examples of dilated convolution for 3×3 kernel. Dilated convolutions are characterized by a space between elements of the kernel, and they increase the receptive field of the network [75].

Encoder	Decoder
2D conv $(6 \times T)$, dilation 1	2D conv $(3 \times T)$, dilation 1
2D conv $(12 \times T)$, dilation 2	2D conv $(6 \times T)$, dilation 1
2D conv $(24 \times T)$, dilation 4	2D conv $(12 \times T)$, dilation 1
2D conv $(48 \times T)$, dilation 8	2D conv $(24 \times T)$, dilation 1
2D conv $(96 \times T)$, dilation 16	2D conv $(48 \times T)$, dilation 1

Table 3.4.: Detailed list of composition network encoder-decoder 2D-convolutions.

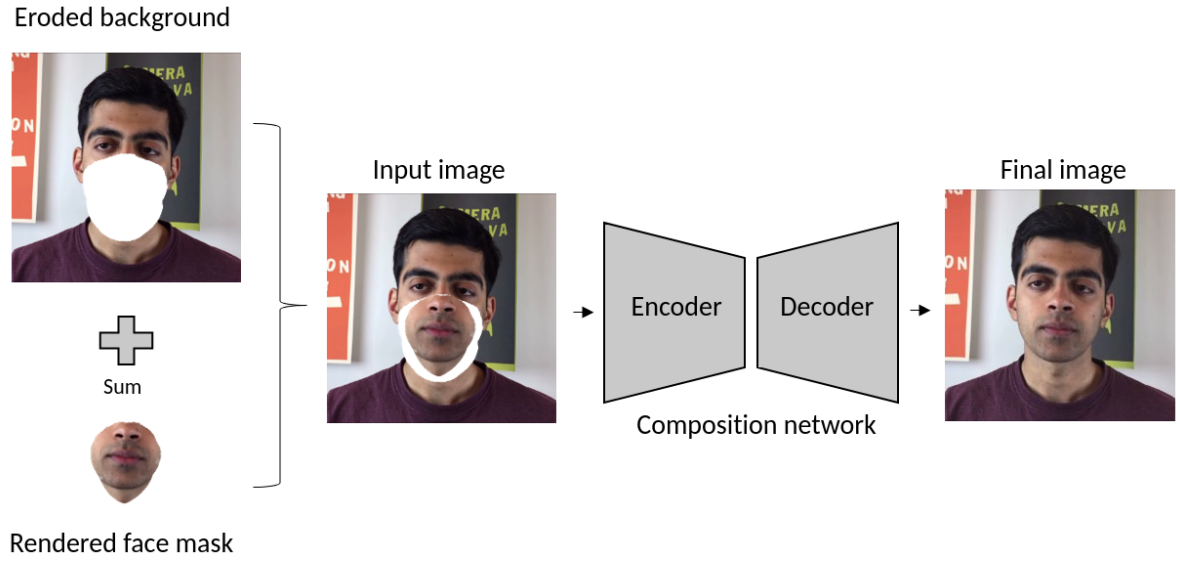


Figure 3.8.: Composition encoder-decoder network overview. The background image is first eroded in order to remove the whole lower part of the face. Next, it is concatenated with rendered face mask creating a 6-channel input image to the encoder. Both networks are built using 2D dilated convolutions architecture, which helps increase the receptive field of the network.

3.7. Loss Functions

The selection of a loss function is one of the most critical aspects of designing a neural network architecture since it can hugely influence the results. This project uses a weighted combination of two loss functions, per pixel L1 distance loss and VGG perceptual loss [76] with additional regularization term on projected audio-features

$$\mathcal{L} = \lambda_{VGGcol} * \mathcal{L}_{VGGcol} + \lambda_{L1col} * \mathcal{L}_{L1col} + \lambda_{L1deform} * \mathcal{L}_{L1deform} + \|\mathbf{b}\|_1 , \tag{3.4}$$

where $\mathbf{b} \in \mathbb{R}^{64}$ is the audio-feature vector described in Section 3.4. L1 loss function is a Manhattan distance between two pixels, it helps to preserve fine details and avoids blurring the image

$$d_1(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i| . \tag{3.5}$$

However, for image generation or classification, it is crucial to capture perceptual differences as well. Especially in the context of audio-driven image generation, because the ground truth is usually shifted at least one pixel due to errors in facial motion predictions. Therefore, the VGG perceptual loss function is used additionally to L1. Instead of a photometric error, it estimates differences w.r.t high-level image features which are generated by a pre-trained

convolutional neural network [76] in this case VGG16 [77]. Finally, L1-regularizer $\|\mathbf{b}\|_1$ was used to avoid overfitting and to enforce a smooth audio-feature vector.

Moreover, a particular weighting strategy of face areas was introduced to help neural networks faster converge in certain regions as well as keep more attention to them during training. According to the used mask, L1 loss is additionally weighted to emphasize the importance of mouth and lips movement. Figure 3.9 presents all types of masks used in the project. The inpainter mask is used in the composition network for eroding the image (Section 3.6).

Mask	Weight
Whole face	32.0
Big mask	128.0
Jaw mask	128.0
Mouth mask	256.0

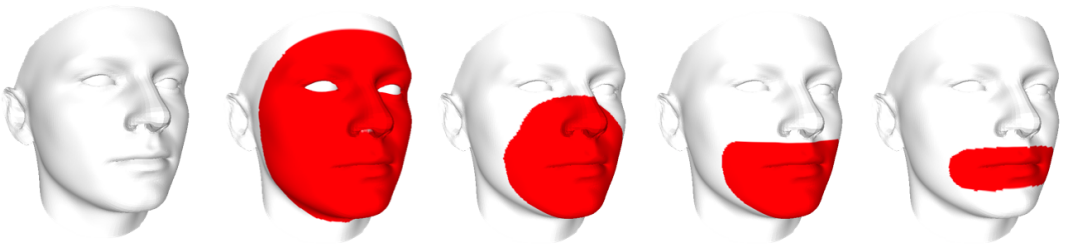
Table 3.5.: Weight distribution for every mask.

During training, each mask from the Table 3.5 is multiplied by a given loss creating a weighted version of L1 loss

$$\mathcal{L}_{L1col} = \sum_{m \in masks} m * d_1(\mathbf{c}, \widehat{\mathbf{c}}), \tag{3.6}$$

$$\mathcal{L}_{L1deform} = \sum_{m \in masks} m * d_1(\mathbf{d}, \widehat{\mathbf{d}}), \tag{3.7}$$

where $\mathbf{c} \in \mathbb{R}^3$ is the color target, $\widehat{\mathbf{c}} \in \mathbb{R}^3$ is the network's color prediction, $\mathbf{d} \in \mathbb{R}^3$ is vertex position target, $\widehat{\mathbf{d}} \in \mathbb{R}^3$ is predicted vertex position.



(a) No mask **(b)** Big mask **(c)** Inpainter mask **(d)** Jaw mask **(e)** Mouth mask

Figure 3.9.: Face masks used for weighting L1 loss function. During rasterization, pixels which constitute marked mesh faces are processed differently (Section 4.1.3).

The composition of perceptual loss and L1 loss produces high-quality images. Using L1 loss networks first find a coarse version of the image and later, during training, VGG loss helps to refine it and learn details.

4. Results

This chapter presents results in contrast to other methods. First, Section 4.1 brings the pipeline setup and the used dataset closer. Next, to have a meaningful comparison framework, Section 4.2 introduces several metrics that constitute the benchmark. Furthermore, a comparison to some state-of-the-art methods is shown in Section 4.3. Finally, in Ablation Study (Section 4.4) network's components are elaborated and their influence on the results.

4.1. Framework Setup

Framework setup is vital for a robust pipeline. In the section, the process of generating data is described. The quality of data is crucial for supervised training and hugely influences the results. Moreover, the structure of dataset and training parameters are covered to give an overview on the project's back-end.

4.1.1. Dataset

Face2Face [8] is a framework for real-time facial animation of a monocular video (e.g., YouTube). The approach is based on 3DMMs optimization using differentiable renderer and face tracker. This project benefits from the face geometry of the model obtained by Face2Face. However, the training is not bound to any framework in particular. Every 3DMMs optimizer for monocular videos should be able to produce sufficient data to train the model. It is speculated[1] that 3DMMs combined with a dedicated parametric teeth model, for instance, Wu et al. [78], would perform much better for image rendering. Most of the 3DMMs do not explicitly model teeth. Therefore, this region is represented by a flat surface, which reduces 3D position variations by one dimension resulting in lower quality of the final image.

4.1.2. Face2Face: 3DMMs Optimization

Thies et al. [8] use 3DMMs created by Blanz and Vetter [17] as optimization framework for facial identity, shape, and reflectance with the following notation to describe the model [8]

$$\mathcal{M}_{geo}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \mathbf{a}_{id} + E_{id} \cdot \boldsymbol{\alpha} + E_{exp} \cdot \boldsymbol{\delta}, \tag{4.1}$$

$$\mathcal{M}_{alb}(\boldsymbol{\beta}) = \mathbf{a}_{alb} + E_{alb} \cdot \boldsymbol{\beta}, \tag{4.2}$$

[1] Those models are not publicly available. Therefore, the experiments cannot be conducted empirically.

where E_{alb}, E_{exp}, E_{id} are eigen basis for albedo, expression and identity. $\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\delta}$ are parameters of the model and $\mathbf{a}_{alb}, \mathbf{a}_{id}$ are average albedo and mesh. The mesh used by [17] has 53K vertices and 106K triangles. Additionally, pose of the faces is described by rotation and translation $\mathbf{R}, \mathbf{t}$ and focal distance of camera $\mathcal{K}$. All parameters are concatenated in one vector $\boldsymbol{\mathcal{P}}$ which is optimized w.r.t each pixel using photo-consistency error [8]

$$E_{col}(\boldsymbol{\mathcal{P}}) = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{p} \in \mathcal{V}} \left\| C_S(\mathbf{p}) - C_I(\mathbf{p}) \right\|_2, \tag{4.3}$$

where C_S is the synthesized image and C_I is the RGB target image, $\mathbf{p} \in \mathcal{V}$ stands for all pixel inside a bounding box of a face. Additionally, to obtain better parameters before photo-consistency optimization and avoid local minima, authors use sparse facial landmarks minimizing the following energy function [8]

$$E_{lan}(\boldsymbol{\mathcal{P}}) = \frac{1}{|\mathcal{F}|} \sum_{\mathbf{f_j} \in \mathcal{F}} w_{conf,j} \left\| \mathbf{f}_j - \Pi(\phi(\mathbf{v}_j) \right\|_2^2, \tag{4.4}$$

where $\Pi(\phi(\mathbf{v}_j))$ is a projection of 3D point $\mathbf{v}_j = \mathcal{M}_{geo}(\boldsymbol{\alpha}, \boldsymbol{\delta}) \in \mathbb{R}^3$ to 2D screen pixel point and $\mathbf{f}_j \in \mathcal{F} \subset \mathbb{R}^2$ is a landmark position. All of that constitute the final energy function for variational optimization [8]

$$E(\boldsymbol{\mathcal{P}}) = w_{col} E_{col}(\boldsymbol{\mathcal{P}}) + w_{lan} E_{lan}(\boldsymbol{\mathcal{P}}) + w_{reg} E_{reg}(\boldsymbol{\mathcal{P}}). \tag{4.5}$$

$E_{reg}(\boldsymbol{\mathcal{P}})$ is parameters regularizer to avoid local minima and produce smooth parameters vector $\boldsymbol{\mathcal{P}}$ close to mean, for more details about energy formulation, please refer to [8].

Using parallel Iteratively Reweighted Least Squares [79] solver implemented on GPU the authors were able to produce precise 3DMMs for any given face using just RGB monocular video.

4.1.3. Processing

In this project, a video of an actor is first processed by Face2Face program to produce a high quality face model. Next, for every frame of the video, geometry of the model is rasterized and 3D points are sampled for every pixel in both states, neutral $\mathbf{a}_{id} + E_{id} \cdot \boldsymbol{\alpha}$ and with additional expressions displacement $\mathbf{a}_{id} + E_{id} \cdot \boldsymbol{\alpha} + E_{exp} \cdot \boldsymbol{\delta}$. Moreover, four different masks are used for sampling (Section 3.7). Figure 4.2 presents the complete set of rendered images for each frame. Except 3D position, pose matrix is stored for the test time rasterization and embedding vector generated using Wav2Lip network for every frame.

To store images rendered in this way and still maintain high precision of 3D points, HDR textures are used. OpenEXR provides a .exr file format for professional image storage with 16-bit floating-point precision. It is a lightweight abstraction for storing HDR textures used in photo-realistic rendering with high-dynamic-range environment light sources.

A dataset prepared in this way is usually heavy. 11000 frames video occupies approx. 40 GB of memory. This project uses a dataset [12] of six actors in total (Figure 4.1). Only Barack Obama's video is in-the-wild. The rest was recorded in a predefined setup where actors read

an English text (some of the actors are native speakers). Each of the videos has 25 frames per second in the resolution of 1280×720. A video is scaled down with aspect ratio conservation and cropped to the final 512×512 size. Finally, the video is converted into a sequence of .png images used as ground truth. A training dataset contains 90% of total frames with reserving the last 10% of the sequence for testing purposes. During training 3000 frames is randomly selected from the whole training dataset.

Actor	Frames	Training	Test	Video time
Ayush	10187	9168	1019	6 min 47 sec
Franziska	7959	7163	796	5 min 18 sec
Jane	5581	5023	558	3 min 43 sec
Jiayi	8917	8025	892	5 min 56 sec
Krista	9664	8698	966	6 min 26 sec
Obama	3145	2831	314	2 min 5 sec

Table 4.1.: Number of frames used to train the model of each actor.

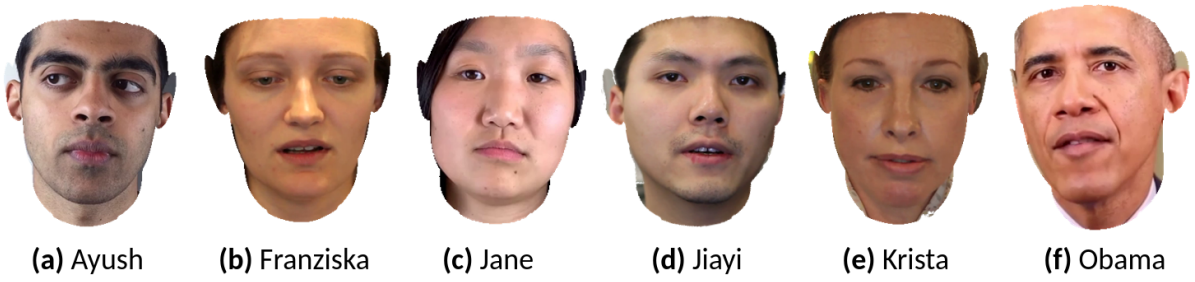


Figure 4.1.: The Neural Voice Puppetry [12] dataset used in this project consists of six actors.

4.1.4. Training

The model is optimized using Adam optimizer [80] with default parameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e^{-8}$). Framework used for training and efficient GPU utilization is PyTorch [81]. All experiments were performed using NVIDIA GeForce GTX 1080 Ti GPU.

Training a fully connected MLP, where the input is a single 3D position/pixel, is time-consuming and very unproductive for an image with $512 \times 512 = 262144$ elements. To alleviate this problem, a convolutional neural network (CNN) is used. Using 2D-convolutions with $kernelsize = 1$ and $stride = 1$, optimization can be done for the whole image at once. However, for $batch = 1$ the CNN model needs approx. 16 GB of memory for an image 512×512 which exceeds 11 GB available memory on GTX1080 Ti. Thus, an image is divided into two smaller ones, which ultimately extends training to twice the time. Another important aspect of optimization is avoiding overfitting. During experiments, many times, the projection network (PN) (Section 3.4) overfitted when it was trained for one actor only. Therefore, in the

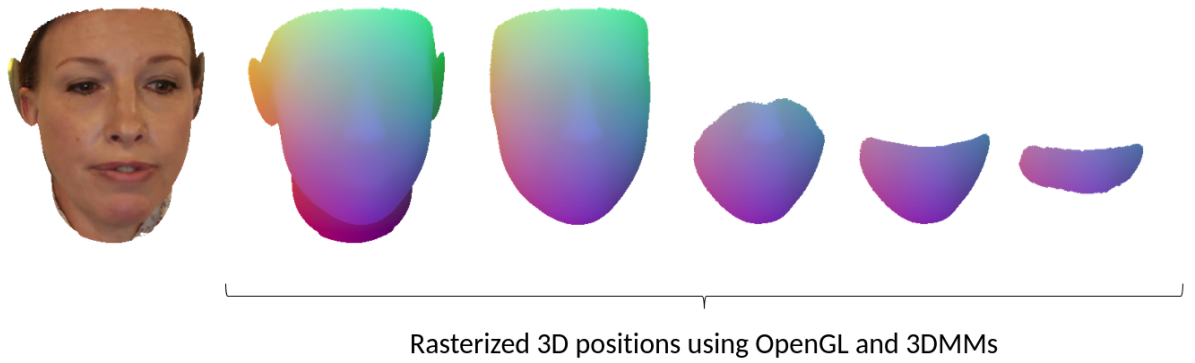


Figure 4.2.: 3DMMs rasterized components for each frame. The resulting sample 3D positions are saved as HDR textures and used to create weighting masks (Section 3.7) and evaluate the loss function for the deformation network. The color presented on the diagram depicts a 3D position for a given pixel.

final implementation, the PN is optimized for at least two actors. The total time of training two actors sequentially is around 72h with 40 epochs, 3000 frames per epoch, and frame window $T = 9$ (Section 3.4). The composition network (Section 3.6) for 3000 frames per epoch and 33 epochs takes 72h per single actor.

4.2. Metrics

A performance evaluation is a crucial part of every scientific project. In this section, several metrics, constituting the benchmark, are presented, which measure two main aspects of the results; image quality and the lip-audio synchronization error.

4.2.1. Image Quality Error

High-quality synthesized face images have to capture many intricate details of the real-world faces. For instance, skin subsurface scattering, facial hair, or teeth structure. Teeth are an example of a step function (Section 3.5.2) which is in particular difficult because of the additional facial motion and infinite frequency content of the function [2] (Section 3.5.2). Therefore, it is important to have a unified strategy to measure the quality.

Metric	Error	Range
SSIM	Local and absolute	[0, 1] (higher better)
PSNR	Absolute	[20, 50] (higher better)
MSE	Absolute	(lower better)

Table 4.2.: Image quality metrics.

Three different measurements are used for image quality Table 4.2. To have an objective

overview, a perceptual structural index and per-pixel distance is calculated separately. Wang et al. [82] introduced the structural similarity index (SSIM), which assess perceptual quality for every patch of pixels 11×11 between two images I_A and I_B using luminance, contrast, and structure. In our case, SSIM measures structural degradation of information for faces, especially the mouth region. It is defined as follows [82]

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4.6}$$

where μ_x, μ_y represent average values for a patch (x, y). σ_x, σ_y are standard deviations for the patch. $C_1 = (K_1 L)^2, C_2 = (K_2 L)^2$ where L represents range of the pixel values[2] and $K_1 \ll 1$ is a small constant [82].

Peak signal-to-noise ratio (PSNR) and mean-square error (MSE) are calculated for each pixel separately in contrast to SSIM. Given two $M \times N$ images I_A and I_B, MSE is calculated as follow

$$MSE(I_A, I_B) = \frac{1}{N * M} \sum_{n=1}^{N} \sum_{m=1}^{M} [I_A(n,m) - I_B(n,m)]^2. \tag{4.7}$$

Finally, PSNR can be calculated using the following formula

$$PSNR(I_A, I_B) = 20 * log_{10}(Max_I) - 10 * log_{10}(MSE(I_A, I_B)), \tag{4.8}$$

where Max_I is a constant value representing a maximum encoded value for a given image, usually it is 255 for 8 bits encoding. The resulting value is expressed in decibel and is in the range between 30 and 50 dB for 8 bits encoding.

4.2.2. Lip Synchronization Error

Besides photo-realistic images, another objective of this work is the high fidelity of lip-audio synchronization. Chung et al. [64] proposed a very convenient way for automatic lip-sync evaluation, which later also was used in [9, 60] to measure performance. To determine lip-sync error in SyncNet [64], the Euclidean distance is measured in 256-dimensional latent space between audio and video embeddings. For every video-feature vector estimated for five consecutive video frames, audio-feature vectors in the range ± 1 second are calculated. Using a sliding window technique, the Euclidean distance is calculated between each audio-video pair. The correct offset is one with the minimum distance. SyncNet achieves 99% accuracy for averaged samples over a clip and 81% for a single sample (0.2s). Except for the distance, Chung et al. [64] developed one more metric called the confidence score of a time offset. It is defined as the difference between the minimum and the median of the distance between feature vectors. High confidence represents good lip-audio synchronization. Based on those findings, two metrics are used to measure the quality of generated audio-driven videos. Similarly to Wav2Lip [9] they are defined as Lip-sync error distance (LSE-D) and Lip-sync

[2] 255 for 8-bit grayscale images.

error confidence (LSE-C). LSE-D is the average Euclidean distance between embeddings over all frames, which describes audio-visual match and LSE-C measures correlation of lips movement and speech [9]. Equipped with the evaluation benchmark, the results can be meaningfully compared with some state-of-the-art solutions. Table 4.3 summarizes the metrics, including a way of interpreting them.

Name	Interpretation
LSE-D	Smaller better
LSE-C	Higher better

Table 4.3.: Lip synchronization error metrics.

4.3. Audio-driven Facial Reenactment

This section presents achieved results with additional comparison to other methods. To the best of our knowledge, two methods currently are state-of-the-art for lip-sync and facial reenactment. Neural Voice Puppetry (NVP) by Thies et al. [12] and Wav2Lip by Prajwal et al. [9]. Both fundamentally different, NVP is based on the 3DMMs immediate geometry that controls the network, whereas, Wav2Lip is a GAN-based approach that generalizes for every subject in contrast to NVP.

4.3.1. Color Network Results

Before introducing the comparison benchmark of all methods, first, self-evaluation is shown. Because the method was trained on short videos, the generalization test can be performed on the test dataset. During the training, 90% of the dataset was used, leaving 10% for the testing purposes.

The following diagrams depict five randomly selected frames from the test sequence. An audio-feature vector of English speech from a testing frame is fed to the network. Next, the generated results are compared with the original video frame. The $\mathcal{L}_2$ distance in RGB color space in the range $[0, 1]$ is calculated between the fake and target frame. The error is presented as a heat map where the colder color (blue) represents low error.

The network synthesizes images of size 512×512. Each figure (4.3, 4.4, 4.5) for self-evaluation comparison contains four rows for five randomly selected frames from the rendered video. The first two rows present the cropped region with fake and target mouth, the third row depicts the $\mathcal{L}_2$ distance in RGB color space. Finally, the last row presents the whole fake frame. The actors speak English; however, only some are native speakers.

The results show that the pipeline is able to render audio-driven photo-realistic images. For Krista and Obama, the fake images seem to best match the ground truth. One of the reasons why the results of Ayush have high error could be the quality of the training video. Please refer to the Appendix to see the results of the remaining actors Figures A.1, A.2, A.3.

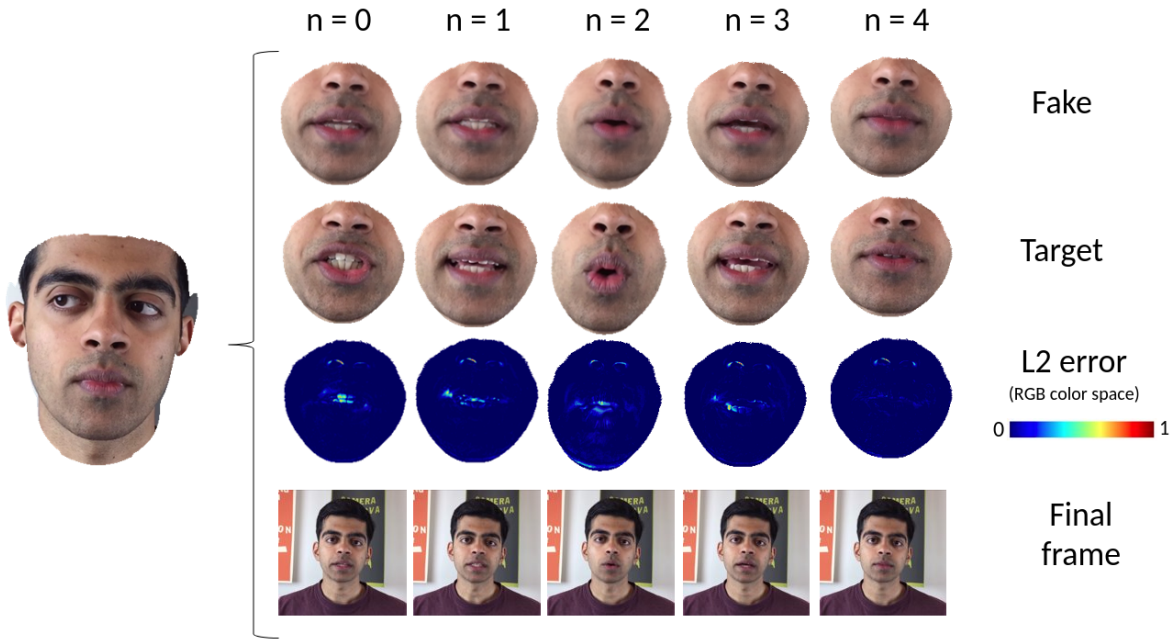


Figure 4.3.: Actor Ayush - five frames, randomly selected from the test set, compared to the ground truth.

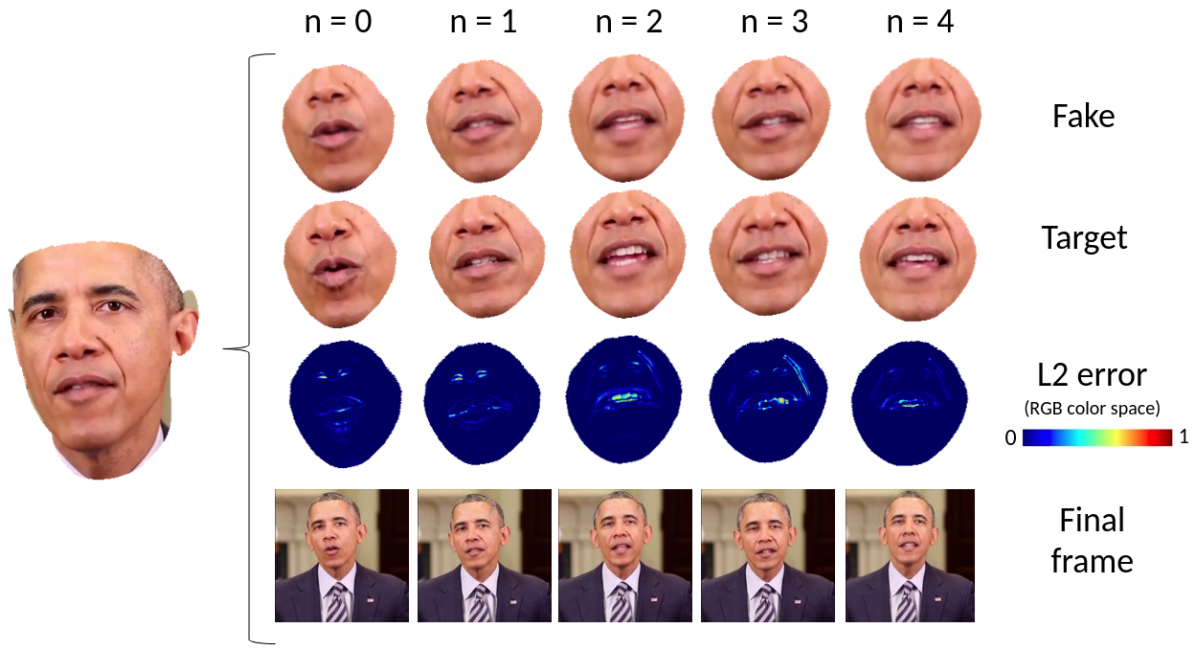


Figure 4.4.: Actor Obama - five frames, randomly selected from the test set, compared to the ground truth.

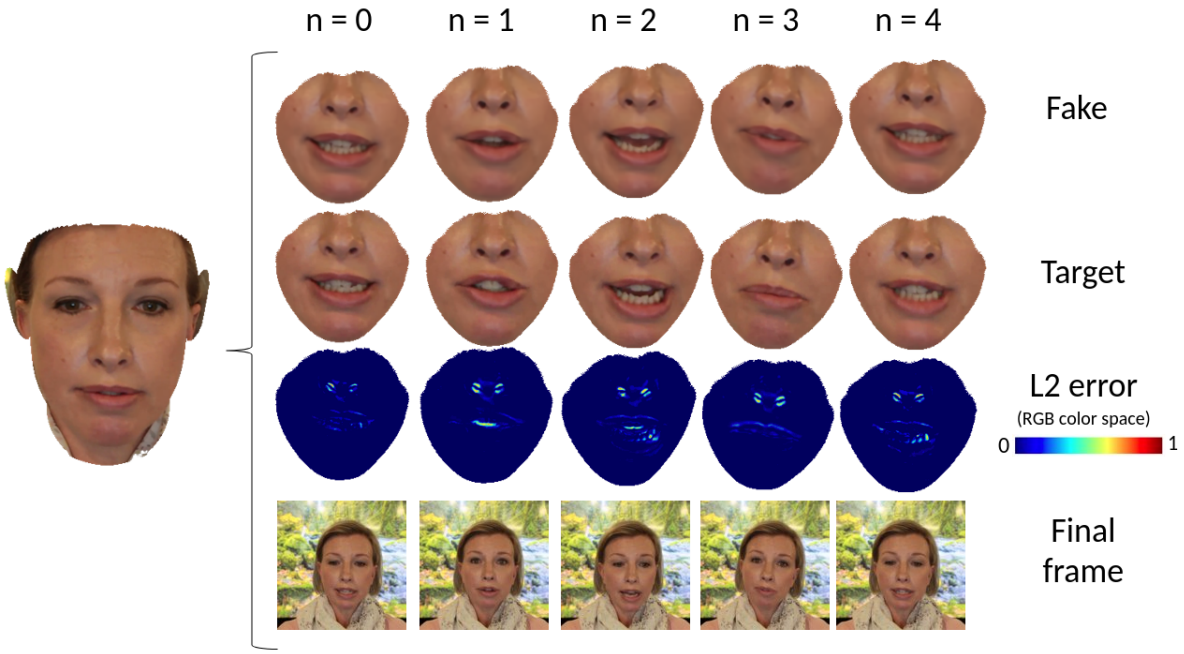
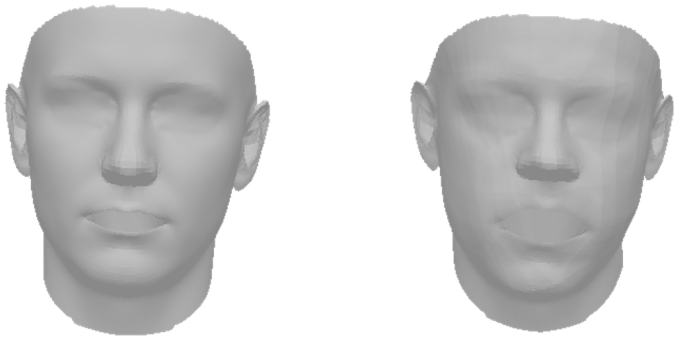


Figure 4.5.: Actor Krista - five frames, randomly selected from the test set, compared to the ground truth.

4.3.2. Deformation Network Results

In the section 3.5.1 the deformation network was described together with its function in the rendering pipeline. During the test time, the 3DMM's mesh is fed to the network, which predicts the geometry based on a given audio-feature vector. Finally, the output is rasterized and passed to the color network.



(a) The input mesh in neutral state **(b)** Mesh's predicted displacement

Figure 4.6.: Actor Ayush - visualization of the input and output mesh during the test time. The deformation network successfully predicts vertex displacements for the face identity and expressions.

Figure 4.6 presents the input and output mesh after predicting the displacement of vertices. The predicted vertex offset was calculated for the actor Ayush. The predicted area is not evenly deformed since the color error guided the optimization. However, it is visible that the network is able to predict the face identity with the additional expression state, which is useful for the color network.

Figure 4.7 shows the mesh deformation states for five randomly selected frames from the test sequence. Despite the fact that the facial expressions were selected to be extreme for the comparison, the difference between predicted face geometry is not clear. It is due to the fact that the video has 25 frames per second, and the correct motion is visible only on a video.

Moreover, the network could not learn a coarse approximation of the teeth structure, even with its immense parameters. It is speculated that a dynamic facial motion during a person's speech is complicated to optimize solely based on the color. As mentioned before, in this situation, a teeth model could be an excellent solution to get a more detailed prediction in this area.

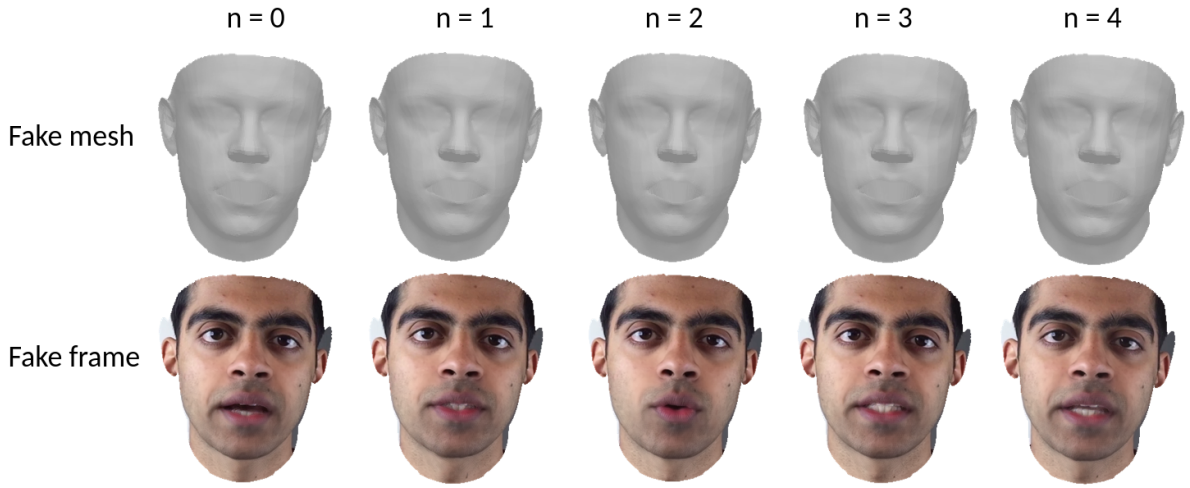


Figure 4.7.: Actor Ayush - predicted mesh representation for randomly selected frames. Even though predicted expressions do not vary between frames greatly, the color network can still make correct predictions.

4.3.3. Corner Cases

The color network is prone to any inconsistencies in the predicted vertex displacement. This is especially important during the test time, where an extra rasterization step is involved in the deformation network's output. In this time, the mesh is first transformed to the expression space spanned by the audio, then projected into the current frame's pose basis, and finally rasterized. Any misalignment will be visible on the final output since the composition network fails to place the distorted image correctly.

Figure 4.8 depicts a situation when for a certain pose transformation, the generated mesh

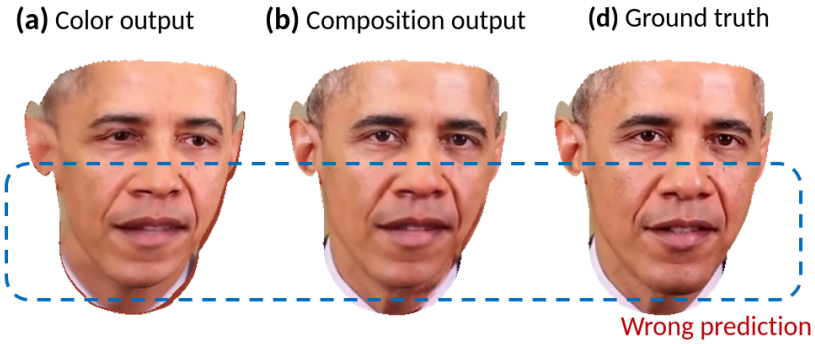


Figure 4.8.: The failure case of the predicted mesh displacement was propagated through the network layers resulting in a wrong image. The actor has artifacts on the lower-left corner of the face because of the wrong geometry, resulting in the double chin effect.

displacement error is propagated through all the network layers. The actor Obama has wrongly deformed the left side of his face. It is visible on a) and b). The network failed to align the correct face identity, generating rendering that does not fit the ground truth. Additionally, Figure 4.9 presents the $\mathcal{L}_2$ distance in RGB range to the ground truth with the outlined misalignment region. An error like that influences the final video reducing the realism of the talking head due to lack of end-to-end optimization.

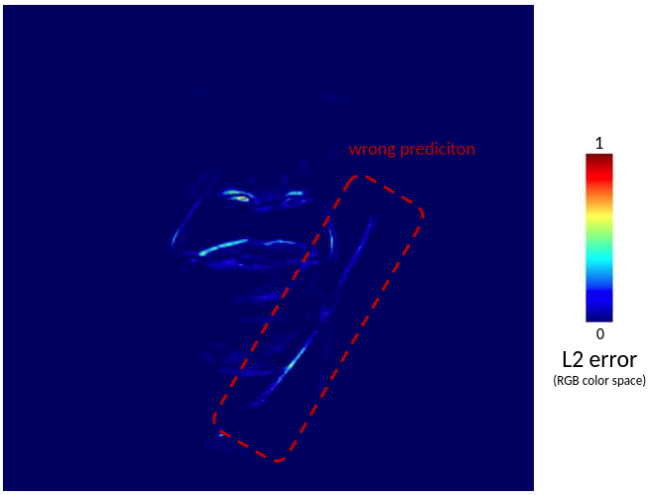
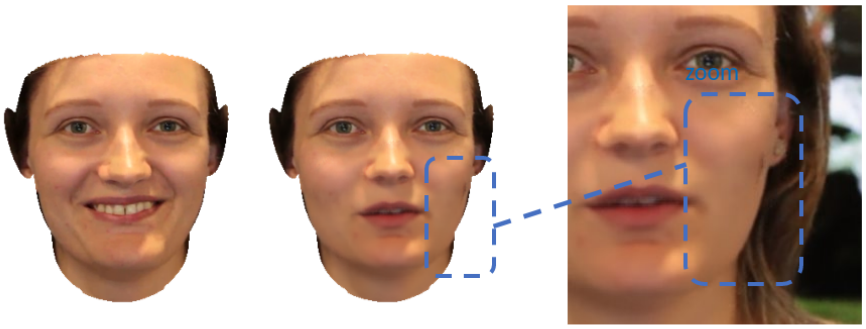


Figure 4.9.: $\mathcal{L}_2$ distance to the ground truth showing the region with the wrong prediction which is affecting the final output. The artifact from Figure 4.8 presented as the heat map of the error.

Another example of a failure case is the situation when the source image (Section 3.6) has different expressions compared to those generated based on the audio input. The composition network fails to properly erode some regions of the face when the angle of the head position

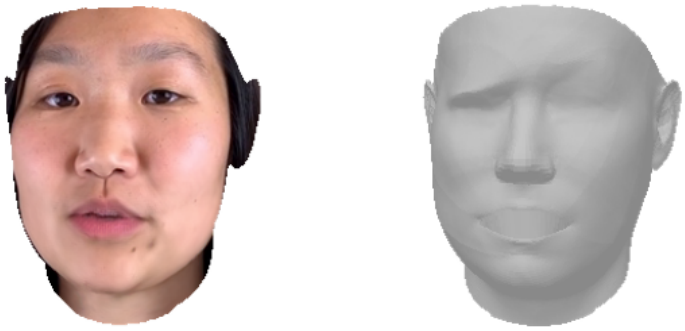
with respect to the camera is steep. Figure 4.10 shows an example where source image expressions overlap with those generated by the pipeline producing artifacts. This situation can be also caused by an outlier in the training sequence, since the smile, or any other short-term expression state, is not learned properly, due to short nature of the video and the different learning objective.



(a) Source frame **(b)** Audio-driven output frame **(c)** Final frame

Figure 4.10.: Example of extreme expressions alignment between the source and target generated based on the audio input.

Finally, the last problem with photometric error optimization is the influence of shadows or hair on the final 3D mesh. If some portion of hair covers parts of a face, those areas' predicted deformation will be wrong. Figure 4.11 presents a situation when the actor Jane's black hair covers part of her upper right face. The 3D mesh has visible deformation in this region.



(a) Fake color **(b)** Fake mesh

Figure 4.11.: Photometric error influences 3D mesh structure. Black hair, which covers some parts of the face, can drive the incorrect local minimum optimization resulting in wrongly predicted displacement.

To summarize, the mistakes like presented above can occur. Problems caused by a small misalignment of geometry or modifying the actor's face with expressions that are notably different from the predicted ones affect the final result.

4.3.4. Lip Synchronization Error Comparison

Lip synchronization is a crucial part of every audio-driven face synthesis. The discrepancy can be spotted almost immediately by a viewer. Therefore, it is important to use a metric that can robustly indicate any inconsistency. For that the benchmark defined in the Section 4.2 based on SyncNet [64] is used. To have a wide overview, first, the ground truth videos are evaluated. This gives a good reference point since the fake videos are generated for the same sequence. All measurements were taken for the 600 frames long test video of each actor.

For the final comparison, some of the state-of-the-art methods were selected. The evaluation was performed for all actors from the dataset based on the test sequences. Additionally, for our and Thies et al. [12] methods, an in-the-wild speech of Bill Gates[3] was taken into consideration. This helps to measure the generalization of both methods outside the dataset. Please refer to the Tables A.2, A.3, A.4 in the appendix for more details. Prajwal et al. have two results in the final Table 4.4. The GAN version has the better visual quality for the price of lip synchronization.

From the results presented in Table 4.4 a few interesting things can be inferred. First, not every actor in the used dataset speaks naturally. The ground truth confidence metric (LSE-C) is low, with average confidence 6.091. Additionally, the variance is 2.82, and the lowest LSE-C score is for Jane (3.471), whereas the highest is for Obama (7.961). It means that the dataset is not well balanced and should be extended with more in-the-wild videos in the future. Moreover, Prajwal et al. [9] results are much better than the ground truth. The Wav2Lip project is using the lip-sync expert based on SyncNet architecture for the GAN's discriminator network. Therefore, the metrics show more significant performance in our benchmark due to the aforementioned reasons about dataset quality and its creation purpose[4].

Method	LSE-D ↓	LSE-C ↑
Ground Truth	7.223	6.091
Thies et al. [12]	9.517	4.131
Suwajanakorn et al. [44]	7.834	6.631
Prajwal et al. [9]	**5.771**	**8.149**
Prajwal et al. GAN [9]	6.230	7.862
Chen et al. [51]	7.467	6.958
Our	7.229	6.090

Table 4.4.: Average lip synchronization error evaluated for the dataset of six actors.

The method with the best score in both metrics is Prajwal et al. [9] which is based on deep GAN architecture. Our method scores almost identically as the ground truth in terms of LSE-D and LSE-C. This is a very good result, which means that audio-driven facial motion is faithfully reproduced for an arbitrary English speech. Only two approaches in the comparison are using intermediate geometry representation of the face, Thies et al. [12] and

[3]Bill Gates, Microsoft Windows 3.1 Introduction, 1995. Source: YouTube.com

[4]The dataset was created primarily for audio-text editing applications [83].

ours. Suwajanakorn et al. [44], and Chen et al. [51] based their solution on sparse mouth landmark representation to generate mouth movement. Both of them get excellent results, which surpass the ground truth.

To summarize, methods that were trained on a large dataset [44, 9, 51] score better than our ground truth because they contain information of many different speaking styles. Therefore, the generated videos are outside single actor learning space like in our case.

4.3.5. Image Quality Error Comparison

Good lip-audio synchronization is only one part of a faithful audio-driven image generation. The synthesized images have to be photo-realistic to mimic the real video fully. This is a difficult problem in the case of faces. Human skin is made of a translucent material that exhibits subsurface scattering events. The physically-based rendering approach models this phenomenon using the bidirectional scattering surface reflectance distribution function (BSSRDF) [2] which is a difficult and time-consuming method. In this project, BSSRDF is indirectly approximated by neural networks. Thus, we can produce almost indistinguishable images without manually crafting necessary materials for the light simulation process. All measurements were taken for the 600 frames long test video of each actor.

Table 4.5 presents the image quality error benchmark. For a given test sequence, each fake frame was compared to the corresponding ground truth frame using three metrics. Our method outperforms all other approaches by a substantial margin.

Method	SSIM	PSNR	MSE
Thies et al. [12]	0.867	25.505	0.004
Prajwal et al. [9]	0.918	31.899	0.002
Prajwal et al. GAN [9]	0.918	31.930	0.002
Our	**0.982**	**37.791**	**0.0002**

Table 4.5.: Average image error evaluated for the dataset of six actors.

4.4. Ablation Studies

The section explains how different components of the pipeline affect the overall results. First, the selection of audio-feature vectors is elaborated, presenting the difference, both in the used architecture and embeddings. Next, the composition network's effect on the image quality is shown in contrast to the pipeline output (Section 3.1). Finally, the importance of the temporal stabilization is presented, each step from the noise injection to sequential is depicted.

4.4.1. Audio Embedding Selection

The discrepancies between models based on Wav2Lip [9] and NVP [12] are noticeable. Figure 4.12 presents five randomly selected frames where the same sound was used to produce

the output. The model which was using NVP audio-features expresses less facial motion compared to the model based on Wav2Lip. The results presented in this subsection are the direct output from the pipeline without using the composition network for mouth to video embedding.

Audio embedding	LSE-D ↓	LSE-C ↑
Prajwal et al. [9]	**6.544**	**4.228**
Thies et al. [12]	11.851	1.225

Table 4.6.: Lip-synchronization error for Wav2Lip and NVP audio-feature vectors.

Table 4.6 presents the lip-sync error for both approaches. The model based on NVP embedding does not align speech with mouth motion well. Despite conditioning latent codes of each frame on the network's output, the pipeline failed to reproduce a realistic talking head image. The LSE-C, which describes confidence, is very low, suggesting wrong offset and incorrect lips shape for a given sound.

Figure 4.12 depicts five corresponding frames from the testing sequence ran on the same audio input. The architecture based on Wav2Lip's audio-features better captures facial motion than the architecture using NVP's embedding.

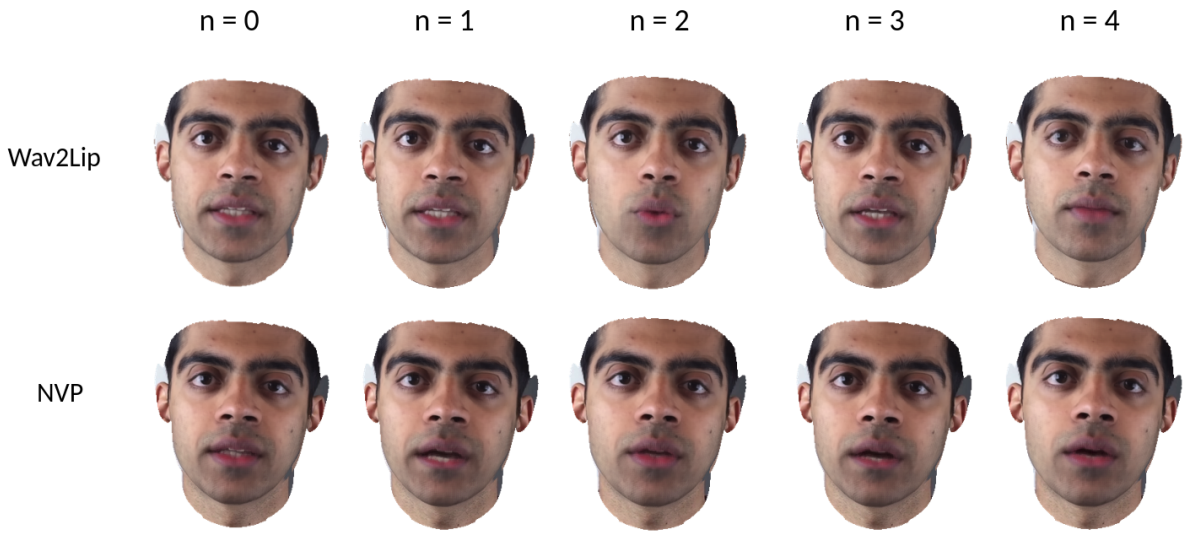


Figure 4.12.: Actor Ayush - discrepancies between Wav2Lip and NVP audio-features for the same sound. Using NVP audio-features produces flat facial motion, which at the end gives an unrealistic looking video. Presented images are the direct output from the color network. Therefore, some artifacts around the face are visible.

4.4.2. Composition Network

The composition network is an essential component of the pipeline. The primary purpose is to properly embed the new mouth into the old source background and improve the mouth interior quality, for instance, a clear teeth structure or a realistic tongue shape. The only input passed to the network is a 6-channel color image (3-channel background plus 3-channel mouth) to make the network speech invariant. For more details, please refer to the Section 3.6. Figure 4.13 presents the actor Obama as the output from the color and composition networks. It can be clearly seen that the composition network quite substantially improves the image quality.

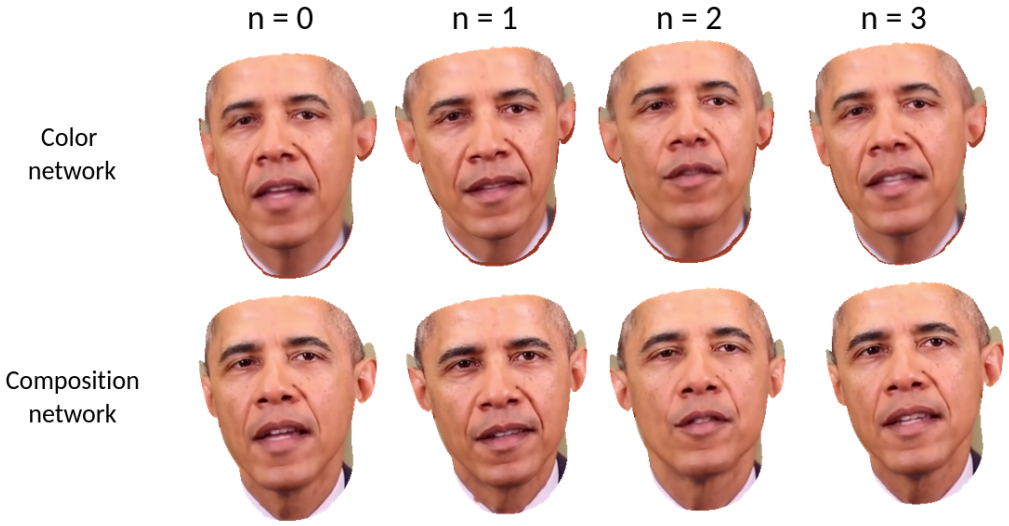


Figure 4.13.: Actor Obama - the comparison of the color network output and the composition network output. The composition network vastly improves the overall quality of the synthesized images.

During the experiments, two different types of 2D convolutions were used, dilated and strided. For more details, please refer to the Section 3.6. Similarly to Neural Voice Puppetry [12], it was confirmed that using dilated convolutions gives much better image quality. However, the time of training and memory consumption of the network is higher compared to strided convolutions. Figure 4.14 presents comparison of those two techniques. The 2D dilated convolution produces a more detailed mouth interior. The facial hair, which exhibits a high-frequency variation of data, is better captured. 2D strided convolution shows a jaggy pattern similar to using the box filter for function reconstruction, suggesting only the nearest-neighbor approximation of the image function.

Network	SSIM	PSNR	MSE
Color	0.952	31.643	0.00070
Composition (dilated 2D conv)	**0.979**	**38.527**	**0.00016**
Composition (strided 2D conv)	0.977	37.970	0.00017

Table 4.7.: Image quality for different pipeline outputs in the project. The best result produces dilated 2D conv network. The color network is the fully connected MLP described in the Section 3.5.2.

Table 4.7 shows the image quality error for three different networks. As a reminder, the color network is a fully connected network based on positional encoding. Together with the composition networks based on two convolution types dilated and strided, the comparison is presented. As it was mentioned previously, the best results are achieved using the dilated 2D convolution. However, the metric error values are very close to each other.

(a) Dilated 2D convolution **(b)** Strided 2D convolution

Figure 4.14.: Comparison of using 2D dilated and strided convolutions in the composition network. Strided convolution produces a jaggy pattern suggesting a poor nearest-neighbor approximation of the image function. On the other hand, dilated convolutions synthesize high-quality images.

4.4.3. Temporal Stabilization

Temporal stabilization is a challenge for neural networks that produce videos. A viewer can immediately spot any inconsistencies. Therefore, this topic was investigated during experiments in this project. The projection network (Section 3.4) is responsible for producing

a temporal coherent audio-feature vector for each frame based on the audio input. However, if it is trained only for one actor, the results are overfitted. Noise injection and sequential training were used to avoid it. The following presents impact of each of those techniques on the final results. The calculated metrics are based on the direct output from the color network since the composition network can influence the measurements.

Id	Actors	Noise Injection	Temporal Weighting
x	1	No	No
y	1	No	Yes ($T = 9$)
z	1	Yes	Yes ($T = 9$)
w	2	Yes	Yes ($T = 9$)

Table 4.8.: Ablation study for temporal coherence. There are four types x, y, z, w which are indicators used for the comparison. The table describes which individual components where included for the particular types.

The final composition consists of four different combinations. Table 4.8 presents selected types. The column "Actors" indicates the number of actors used for the sequential training. $T = 9$ in the "Temporal Weighting" column is the length of the used window (Section 3.4) and the rest is self-explanatory. The final configuration used in the project is w.

The types described in Table 4.8 are used as identifiers for a given combination in the following comparison tables. The first benchmark contains the image quality error introduced by each of the setups (Table 4.9). Interestingly, the best results were achieved for those which does not inject noise. However, the visual effect is still the best for w, which has the least amount of artifacts.

Id	SSIM	PSNR	MSE
x	0.9861	39.431	0.000132
y	**0.9867**	**39.747**	**0.000123**
z	0.9858	39.283	0.000139
w	0.9863	39.537	0.000131

Table 4.9.: Actor Ayush - Ablation study for image quality error for temporal coherence comparison. All four combinations were evaluated in the context of introduced image quality error.

Next, the lip synchronization error was evaluated for two different audio sources. The first audio signal is from Ayush's test sequence, whereas the second audio comes from the in-the-wild speech of Bill Gates. According to the benchmark, the best results are achieved by the explicit projection without the temporal coherence stabilization. However, like in the image quality comparison, this result is not reliable since the artifacts for the combination x are much greater than any else. Moreover, x suffers from a deficient generalization level, which is unacceptable in audio-driven applications.

Id	LSE-D ↓	LSE-C ↑
x	**5.755**	**4.578**
y	5.644	4.479
z	6.099	3.777
w	5.827	4.173

Table 4.10.: Actor Ayush - Ablation study for lip synchronization error for temporal coherence comparison evaluated on the test sequence audio. The presented results are giving the misleading notion that x gives the best results, whereas this combination does not generalize well to other audio inputs.

Id	LSE-D ↓	LSE-C ↑
x	**7.110**	**8.186**
y	7.523	7.448
z	7.955	6.620
w	7.649	7.064

Table 4.11.: Actor Ayush - Ablation study for lip synchronization error for temporal coherence comparison evaluated on the Bill Gates sequence audio. An arbitrary English speech does not produce plausible results for the x, and the generated video contains many artifacts that spoil the general impression.

The results of the temporal stabilization effect are giving misleading information. The best way to evaluate them correctly is conducting user studies similarly to Thies et al. [12]. Lip synchronization error does not account for small artifacts during the facial motions, especially for teeth. The same goes for the image quality error. The visible inconsistencies are averaged over the testing sequences and do not have an impact on the metric. Therefore, more elaborated metrics have to be developed for future work, including user studies for consistent and accurate measurements.

5. Limitations

Even though the method produces good results, there are three main limitations that have to be addressed in the future. The first is significantly reduced mouth geometry, which does not include teeth. It is speculated that using a dedicated mouth interior model would improve the training process. An even bigger limitation is the lack of generalization across different actors or scenes. The current implementation needs to retrain the model every time the scene or actor changes.

5.1. Limited Mouth Geometry

3DMMs do not explicitly model the mouth interior, which is a problem in the context of this work. Having detailed geometry of teeth would better capture facial motion during speech. However, it still hugely depends on the speaking style of an actor. Nonetheless, the current approach is about connecting two lips together, creating a closed mouth. Figure 5.1 presents a workaround used in this project. The mouth is sewn up. However, the amount of vertices is not changed. Thus, eigenbasis still holds, and all transformations are valid.

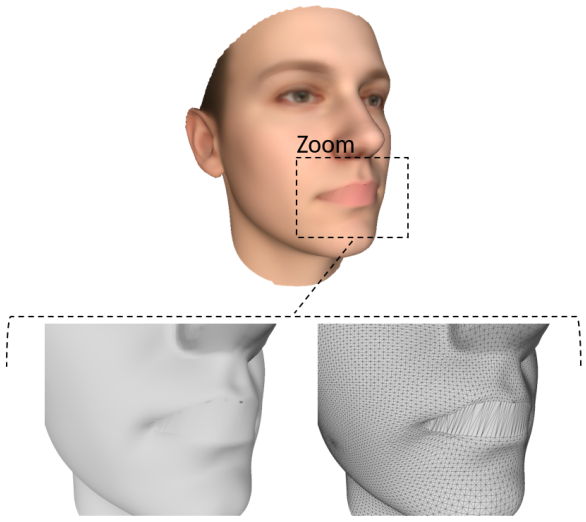


Figure 5.1.: 3DMM's simplified mouth region used in the project. The mesh of the 3DMM was adjusted for the purpose of this project. For instance, the mouth region is closed for the geometry sampling and positional encoding since this is the project's main objective.

One solution for this problem would be using an explicit model for the mouth interior. Wu et al. [78] presented an exact teeth model based on high-resolution plaster cast 3D scans of teeth. Moreover, they show that using a single monocular video frame is sufficient to reconstruct upper and lower teeth. A teeth model obtained in that way could be incorporated into existing 3DMMs for faces resulting in higher variation in data. This is particularly useful for the audio-driven approach described in this project. Figure 5.2 shows an example of such integration.

Unfortunately, the model created by Wu et al. [78] is not publicly available, which makes it difficult to prove the stated thesis. Creating a teeth model is an expensive and time-consuming process. Thus, companies like Disney prefer to keep it for their internal use only.

Combining many parametric models is the base idea of Medusa[1] software by DisneyResearch, where except the aforementioned teeth model, they use eyes, skin, hair, or jaw. The latter by Zoss et al. [84] would also be a potential improvement in audio-driven video generation since talking is inherently connected with jaw motion. However, their system requires a specialized data capture system with four marker cubes on a steel ping attached to teeth using dental glue and cannot be recovered from a monocular video frame.

Figure 5.2.: Integration of parametric teeth model into 3DMMs. Parametric face models do not model teeth explicitly. However, there is a possibility to combine a teeth model and 3DMMs into one detailed model which in theory would improve the results. Image from [78].

[1]Link: https://studios.disneyresearch.com/digital-humans/

5.2. Generalization

Generalization is the biggest downside of the approach used in this work. Changing actors, using novel views, scenes or lightning always requires training the whole pipeline again. This solution does not scale well compared to methods based on GANs (Section 2.5).

For rendering from novel views under changing lighting conditions, a potential solution would be to combine the audio-driven approach with neural radiance fields and 3DMMs. An example of this work was presented by Gafni et al. [36]. However, they based their pipeline on [23] which does not model the scene relighting or materials. Therefore, it is not possible to synthesize a new image under changed lighting parameters. 3DMMs would need to be integrated into NeRV [35] approach, an extension of NeRF with the additional estimation of materials and novel illuminance known from reversed rendering solutions. Combining audio-driven 3DMMs with NeRV could be an answer for the lighting changing problem and novel view rendering. Nevertheless, to the best of our knowledge, the problem of switching scenes for neural radiance fields without retraining is still unsolved.

Unfortunately, replacing actors is also a difficult problem to solve since the network is designed in a way to learn a specific speaking style of a given person. One approach would be to learn each actor's speaking style separately and then use bagging for generalization. Bagging is a technique for joint inference from a committee of classifiers [68], which would be a set of trained models of each actor. Alternatively, one could use transfer learning (TL) methods [85]. Those methods can transfer knowledge between related domains. For instance, each actor would be one domain, and the network can generalize across all domains.

5.3. Future Work

To summarize, the proposed method exhibits three main problems. The first one is the limited level of details for mouth geometry. The second one is scene limitations, including novel camera and lighting parameters, and the third is the lack of generalization across new actors. Future work would investigate more of those problems, especially scene manipulation and relighting since it is an essential aspect for many applications like dubbing or digital resurrection.

Finding a way to generalize the network additionally using GANs is an exciting challenge. There is much promise in the recent studies about combining deep GANs with 3DMMs [59]. Thus, maybe the future research will focus more on exploring latent space of powerful network like StyleGAN [26, 27] bringing those two methods into one universal face renderer with rich facial features and photo-realistic appearance, however, still easily controllable.

6. Conclusion

The architecture presented in this project proves that subject-specific rendering of photo-realistic images can be achieved using a small fully-connected neural network with additional audio-features extracted from an arbitrary English speech. Recent discoveries in the field of positional encoding for image generation were leveraged to achieve high-quality renderings. The composition network improves on that seamlessly embedding the rendered face into the final frame. Moreover, our method qualitative and quantitative is close to or better than some state-of-the-art solutions. In particular, the image quality results outperform other methods. In contrast, lip synchronization error is as low as the ground truth of the used dataset. Therefore, the method does not sacrifice photorealism over facial motion generation. However, the limitation, compared to the methods like Wav2Lip, is the subject-specific nature of the project. Each actor needs a separate neural model. This is time-consuming and still restricted to one scene only.

To summarize, our solution creates a holistic framework that is able to produce high-quality portrait videos driven only by an arbitrary English speech. This novel approach has great potential for the future since there is plenty of room for new progress in this field. Recent improvements of positional encoding [25, 86] give very good basis for further research. Neural rendering is young and a still-evolving discipline where new ideas are developed every day.

A. Appendix

This section contains the rest of the results for each actor. Additionally, a reader can find detailed tables with error metrics for both image quality and lip synchronization benchmarks.

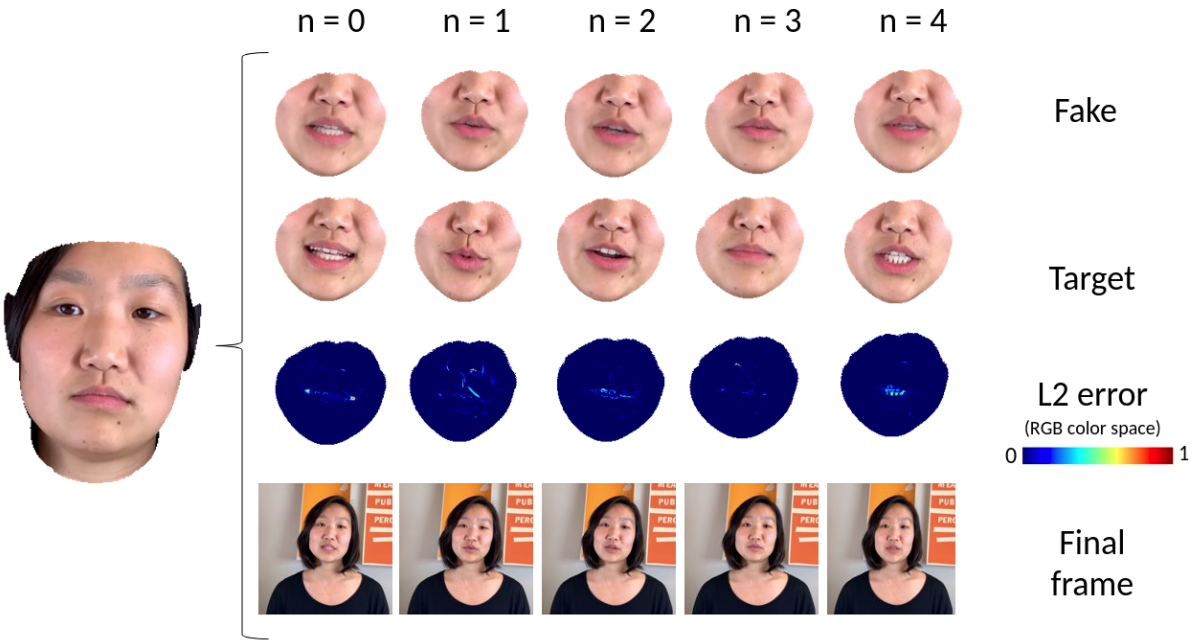


Figure A.1.: Actor Jane - five frames, randomly selected from the test set, compared to the ground truth.

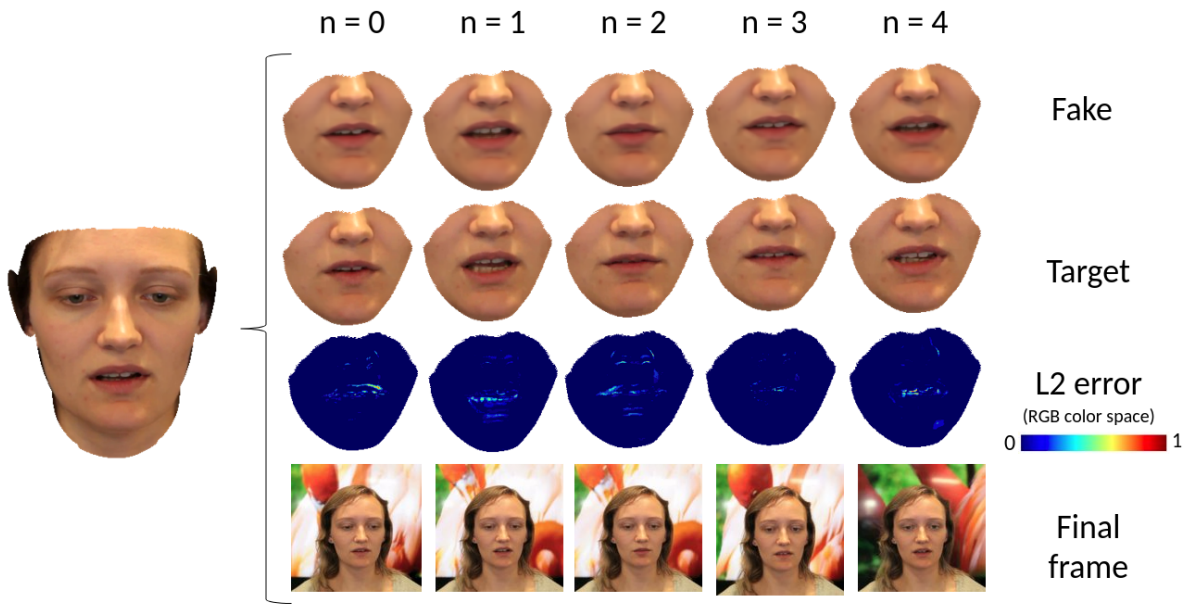


Figure A.2.: Actor Franziska - five frames, randomly selected from the test set, compared to the ground truth.

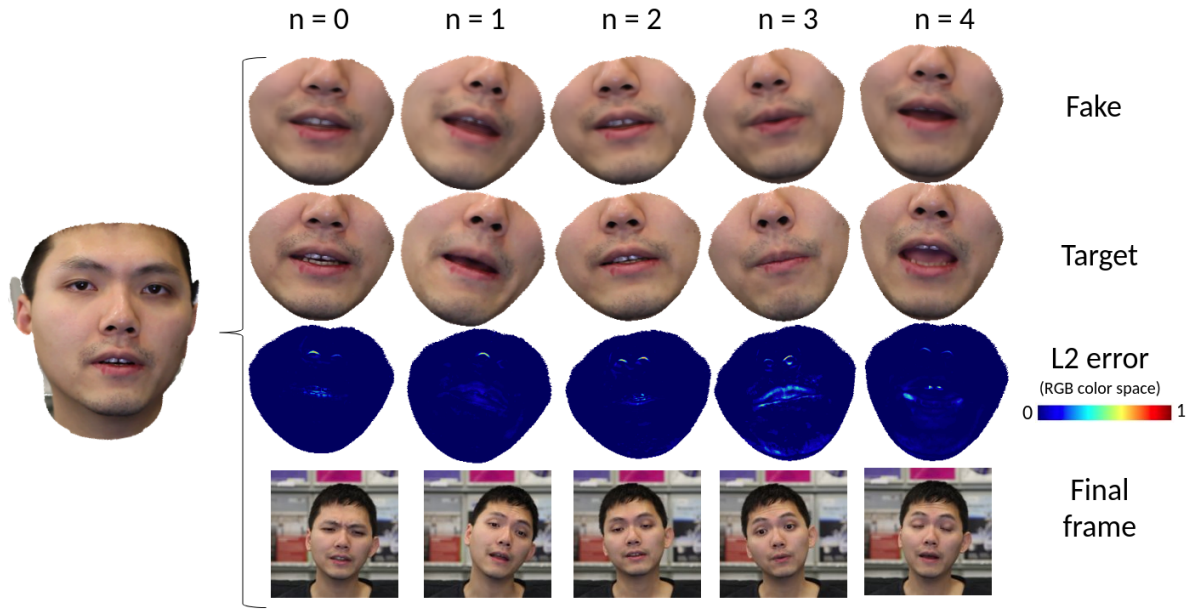


Figure A.3.: Actor Jiayi - five frames, randomly selected from the test set, compared to the ground truth.

Actor	**Our**			**NVP** [12]			**Wav2Lip** [9]			**Wav2Lip GAN** [9]		
	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE
Obama	0.983	38.080	0.00018	0.852	24.814	0.00417	0.918	30.470	0.00108	0.917	30.470	0.00109
Ayush	0.979	38.527	0.00016	0.847	25.174	0.00312	0.922	32.575	0.00063	0.922	32.654	0.00063
Krista	0.988	40.935	0.00009	0.875	26.069	0.00264	0.917	32.341	0.00093	0.917	32.350	0.00092
Jane	0.987	38.965	0.00015	0	0	0	0.935	32.821	0.00068	0.935	32.803	0.00069
Franziska	0.980	34.619	0.00041	0.864	25.791	0.00905	0.895	31.693	0.00676	0.894	31.623	0.00677
Jiayi	0.974	35.623	0.00031	0.899	25.681	0.00357	0.925	31.492	0.00269	0.925	31.680	0.00269

Table A.1.: Detailed table of image quality error across the dataset. Our method outperforms all the other methods in terms of image quality for all three metrics. Please note that the actor Jane for Thies et al. [12] method was excluded from the comparison.

Actor	**Dataset**		**Our**		**Prajwal et al.** [9]		**Prajwal et al. GAN** [9]		**Thies et al.** [12]		**Chen et al.** [51]	
	LSE-D	LSE-C	LSE-D	LSE-C	LSE-D	LSE-C	LSE-D	LSE-C	LSE-D	LSE-C	LSE-D	LSE-C
Obama	6.906	7.961	8.389	5.860	5.411	9.930	5.983	9.579	9.149	4.964	8.114	5.961
Ayush	5.746	5.188	6.082	4.161	4.765	6.365	5.174	6.433	6.921	2.621	7.897	6.395
Krista	7.109	7.792	7.305	6.976	5.683	9.594	6.288	9.201	9.191	4.392	7.122	7.297
Jane	7.669	3.471	7.108	3.313	6.338	5.176	6.427	5.232	0	0	7.511	6.858
Franziska	8.174	6.261	8.279	5.581	6.270	9.351	6.956	8.674	10.030	3.215	6.905	7.912
Jiayi	7.771	5.870	6.831	5.259	6.160	8.481	6.553	8.055	8.709	4.791	7.255	7.324

Table A.2.: Detailed table of lip synchronization error across the dataset. Please note that Prajwal et al. [9] performs so well because the discriminator used in the pipeline is based on the lip-sync expert. Therefore, the whole training process aims to minimize this objective. Moreover, the dataset quality problem described in the section 4.3.4 emphasizes the results. Additionally, the actor Jane for Thies et al. [12] method was excluded from the comparison.

Actor	LSE-D ↓	LSE-C ↑
Obama 1	8.409	6.351
Obama 2	8.115	6.500
Obama 3	7.960	6.850
Obama 4	6.850	6.822

Table A.3.: Lip synchronization error for Suwajanakorn et al. [44]. Four videos with the actor Obama were generated and the lip-sync error was evaluated.

Actor	**Our**		**Thies et al.** [12]	
	LSE-D	LSE-C	LSE-D	LSE-C
Obama	7.857	6.809	9.713	4.854
Ayush	7.574	7.128	11.839	2.997
Krista	7.639	7.438	10.098	4.623
Franziska	8.281	5.995	10.265	3.769
Jiayi	7.350	7.322	9.252	5.075

Table A.4.: Lip synchronization error for Bill Gates speech. Comparison to Thies et al. [12] method for given actors. Our method better generalizes to in-the-wild English speech in all cases.

List of Figures

List of Tables

Bibliography

[1] J. T. Kajiya. "The Rendering Equation". In: SIGGRAPH '86 (1986), pp. 143–150. DOI: 10.1145/15922.15902. URL: https://doi.org/10.1145/15922.15902.

[2] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN: 0128006455.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: NIPS'14 (2014), pp. 2672–2680.

[4] A. Brock, J. Donahue, and K. Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *CoRR* abs/1809.11096 (2018). arXiv: 1809.11096. URL: http://arxiv.org/abs/1809.11096.

[5] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *CoRR* abs/1703.10593 (2017). arXiv: 1703.10593. URL: http://arxiv.org/abs/1703.10593.

[6] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, and M. Zollhöfer. "State of the Art on Neural Rendering". In: *EG* (2020).

[7] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. "FaceVR: Real-Time Gaze-Aware Facial Reenactment in Virtual Reality". In: 37.2 (June 2018). ISSN: 0730-0301. DOI: 10.1145/3182644. URL: https://doi.org/10.1145/3182644.

[8] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. "Face2Face: Real-time Face Capture and Reenactment of RGB Videos". In: (2016).

[9] K. R. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar. "A Lip Sync Expert Is All You Need for Speech to Lip Generation In the Wild". In: MM '20 (2020), pp. 484–492. DOI: 10.1145/3394171.3413532. URL: https://doi.org/10.1145/3394171.3413532.

[10] H. Kim, M. Elgharib, M. Zollhöfer, H.-P. Seidel, T. Beeler, C. Richardt, and C. Theobalt. "Neural Style-Preserving Visual Dubbing". In: 38.6 (2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356500. URL: https://doi.org/10.1145/3355089.3356500.

[11] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. "FaceForensics++: Learning to Detect Manipulated Facial Images". In: (2019).

[12] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, and M. Nießner. "Neural Voice Puppetry: Audio-driven Facial Reenactment". In: *ECCV 2020* (2020).

[13] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen. "Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion". In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073658. URL: https://doi.org/10.1145/3072959.3073658.

[14] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black. "Capture, Learning, and Synthesis of 3D Speaking Styles". In: (2019). arXiv: 1905.03079 [cs.CV].

[15] H. X. Pham, S. Cheung, and V. Pavlovic. "Speech-Driven 3D Facial Animation with Implicit Emotional Awareness: A Deep Learning Approach". In: (2017), pp. 2328–2336. DOI: 10.1109/CVPRW.2017.287.

[16] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. "A 3D Face Model for Pose and Illumination Invariant Face Recognition". In: *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS) for Security, Safety and Monitoring in Smart Environments* (2009).

[17] V. Blanz and T. Vetter. "A Morphable Model for the Synthesis of 3D Faces". In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194. ISBN: 0201485605. DOI: 10.1145/311535.311556. URL: https://doi.org/10.1145/311535.311556.

[18] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. "Learning a model of facial shape and expression from 4D scans". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36.6 (2017), 194:1–194:17. URL: https://doi.org/10.1145/3130800.3130813.

[19] B. Egger, W. A. P. Smith, A. Tewari, S. Wuhrer, M. Zollhöfer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, C. Theobalt, V. Blanz, and T. Vetter. "3D Morphable Face Models - Past, Present and Future". In: *CoRR* abs/1909.01815 (2019). arXiv: 1909.01815. URL: http://arxiv.org/abs/1909.01815.

[20] J. Thies, M. Zollhöfer, and M. Nießner. "Deferred Neural Rendering: Image Synthesis using Neural Textures". In: *ACM Transactions on Graphics 2019 (TOG)* (2019).

[21] P. K R, R. Mukhopadhyay, J. Philip, A. Jha, V. Namboodiri, and C. V. Jawahar. "Towards Automatic Face-to-Face Translation". In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. Nice, France: Association for Computing Machinery, 2019, pp. 1428–1436. ISBN: 9781450368896. DOI: 10.1145/3343031.3351066. URL: https://doi.org/10.1145/3343031.3351066.

[22] K. Vougioukas, S. Petridis, and M. Pantic. "Realistic Speech-Driven Facial Animation with GANs". In: *CoRR* abs/1906.06337 (2019). arXiv: 1906.06337. URL: http://arxiv.org/abs/1906.06337.

[23] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: (2020).

[24] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. "pixelNeRF: Neural Radiance Fields from One or Few Images". In: (2020). arXiv: 2012.02190 [cs.CV].

[25] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: (2020). arXiv: 2006.10739 [cs.CV].

[26] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948. URL: http://arxiv.org/abs/1812.04948.

[27] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *CoRR* abs/1912.04958 (2019). arXiv: 1912.04958. URL: http://arxiv.org/abs/1912.04958.

[28] E. Veach. "Robust Monte Carlo Methods for Light Transport Simulation". In: (1998). AAI9837162.

[29] P. Dutré. *Global Illumination Compendium*. 2003.

[30] B. Burley. "Physically-Based Shading at Disney". In: (2012).

[31] C. d. R. Sebastien Lagarde. "Moving Frostbite to Physically Based Rendering 3.0". In: SIGGRAPH '2014 (2015).

[32] B. Karis. "Real Shading in Unreal Engine 4". In: SIGGRAPH '2013 (2013).

[33] B. Hu, J. Guo, Y. Chen, M. Li, and Y. Guo. "DeepBRDF: A Deep Representation for Manipulating Measured BRDF". In: *Computer Graphics Forum* 39 (2020).

[34] M. Lagunas, S. Malpica, A. Serrano, E. Garces, D. Gutierrez, and B. Masia. "A Similarity Measure for Material Appearance". In: *ACM Transactions on Graphics (SIGGRAPH 2019)* 38.4 (2019).

[35] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. "NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis". In: (2020). arXiv: 2012.03927 [cs.CV].

[36] G. Gafni, J. Thies, M. Zollöfer, and M. Nießner. "Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction". In: (2020).

[37] J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Nießner. "Image-guided Neural Object Rendering". In: (2020). URL: https://openreview.net/forum?id=Hyg9anEFPS.

[38] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. "DeepVoxels: Learning Persistent 3D Feature Embeddings". In: *CoRR* abs/1812.01024 (2018). arXiv: 1812.01024. URL: http://arxiv.org/abs/1812.01024.

[39] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. "On the Spectral Bias of Neural Networks". In: (2019). arXiv: 1806.08734 [stat.ML].

[40] Y. Sato, M. Wheeler, and K. Ikeuchi. "Object Shape and Reflectance Modeling from Observation". In: *proceedings of ACM Siggraph '97 (Computer Graphics)* (Apr. 2001). DOI: 10.1007/978-1-4615-0797-0_4.

[41] R. Ramamoorthi and P. Hanrahan. "A Signal-Processing Framework for Inverse Rendering". In: SIGGRAPH '01 (2001), pp. 117–128. DOI: 10.1145/383259.383271. URL: https://doi.org/10.1145/383259.383271.

[42] E. E. Catmull. "A Subdivision Algorithm for Computer Display of Curved Surfaces." AAI7504786. PhD thesis. 1974.

[43] M. Brand. "Voice Puppetry". In: SIGGRAPH '99 (1999), pp. 21–28. DOI: 10.1145/311535.311537. URL: https://doi.org/10.1145/311535.311537.

[44] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman. "Synthesizing Obama: Learning Lip Sync from Audio". In: 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073640. URL: https://doi.org/10.1145/3072959.3073640.

[45] B. Fan, L. Wang, F. Soong, and L. Xie. "Photo-real talking head with deep bidirectional LSTM". In: (Apr. 2015). DOI: 10.1109/ICASSP.2015.7178899.

[46] P. Lata, C. Kiran, B. Tungathurthi, H. Ram, R. M. R. Hanmanth, D. Govardhan, and D. Reddy. "Facial recognition using eigenfaces by PCA". In: *SHORT PAPER International Journal of Recent Trends in Engineering* 1 (Apr. 2009).

[47] J. J. Atick, P. A. Griffin, and A. N. Redlich. "Statistical Approach to Shape from Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two-Dimensional Images". In: *Neural Computation* 8.6 (1996), pp. 1321–1340. DOI: 10.1162/neco.1996.8.6.1321.

[48] T. Yenamandra, A. Tewari, F. Bernard, H.-P. Seidel, M. Elgharib, D. Cremers, and C. Theobalt. "i3DMM: Deep Implicit 3D Morphable Model of Human Heads". In: (Nov. 2020).

[49] G. Pulford. "The Viterbi algorithm". In: (Jan. 2006), pp. 53–65. DOI: 10.1049/ic:20060556.

[50] Pengyu Hong, Zhen Wen, and T. S. Huang. "Real-time speech-driven face animation with expressions using neural networks". In: *IEEE Transactions on Neural Networks* 13.4 (2002), pp. 916–927. DOI: 10.1109/TNN.2002.1021892.

[51] L. Chen, R. K. Maddox, Z. Duan, and C. Xu. "Hierarchical Cross-Modal Talking Face Generationwith Dynamic Pixel-Wise Loss". In: *CoRR* abs/1905.03820 (2019). arXiv: 1905.03820. URL: http://arxiv.org/abs/1905.03820.

[52] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active Appearance Models". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.6 (June 2001), pp. 681–685. ISSN: 0162-8828. DOI: 10.1109/34.927467. URL: https://doi.org/10.1109/34.927467.

[53] Y. Zhou, J. Deng, I. Kotsia, and S. Zafeiriou. "Dense 3D Face Decoding over 2500FPS: Joint Texture & Shape Convolutional Mesh Decoders". In: *CoRR* abs/1904.03525 (2019). arXiv: 1904.03525. URL: http://arxiv.org/abs/1904.03525.

[54] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition". In: *CoRR* abs/1412.5567 (2014). arXiv: 1412.5567. URL: http://arxiv.org/abs/1412.5567.

[55] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. "FaceWarehouse: A 3D Facial Expression Database for Visual Computing". In: *IEEE transactions on visualization and computer graphics* 20 (Mar. 2014), pp. 413–25. DOI: 10.1109/TVCG.2013.249.

[56] P. Tzirakis, A. Papaioannou, A. Lattas, M. Tarasiou, B. W. Schuller, and S. Zafeiriou. "Synthesising 3D Facial Motion from "In-the-Wild" Speech". In: *CoRR* abs/1904.07002 (2019). arXiv: 1904.07002. URL: http://arxiv.org/abs/1904.07002.

[57] J. S. Chung, A. Jamaludin, and A. Zisserman. "You said that?" In: *CoRR* abs/1705.02966 (2017). arXiv: 1705.02966. URL: http://arxiv.org/abs/1705.02966.

[58] K. Vougioukas, S. Petridis, and M. Pantic. "End-to-End Speech-Driven Facial Animation with Temporal GANs". In: (2018). arXiv: 1805.09313 [eess.AS].

[59] A. Tewari, M. Elgharib, G. Bharaj, F. Bernard, H.-P. Seidel, P. Pérez, M. Zollhöfer, and C. Theobalt. "StyleRig: Rigging StyleGAN for 3D Control over Portrait Images". In: (2020). arXiv: 2004.00121 [cs.CV].

[60] F. Meissen. "AudioStyleNet - Controlling StyleGAN through Audio." 2020.

[61] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208.

[62] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[63] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *CoRR* abs/1901.05103 (2019). arXiv: 1901.05103. URL: http://arxiv.org/abs/1901.05103.

[64] J. S. Chung and A. Zisserman. "Out of Time: Automated Lip Sync in the Wild". In: (Mar. 2017), pp. 251–263. DOI: 10.1007/978-3-319-54427-4_19.

[65] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011* 15 (Jan. 2011), pp. 315–323.

[66] T. Afouras, J. S. Chung, A. W. Senior, O. Vinyals, and A. Zisserman. "Deep Audio-Visual Speech Recognition". In: *CoRR* abs/1809.02108 (2018). arXiv: 1809.02108. URL: http://arxiv.org/abs/1809.02108.

[67] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. "Self-Attention Generative Adversarial Networks". In: (2019). arXiv: 1805.08318 [stat.ML].

[68] C. M. Bishop. "Training with Noise is Equivalent to Tikhonov Regularization". In: *Neural Computation* 7.1 (1995), pp. 108–116. DOI: 10.1162/neco.1995.7.1.108. eprint: https://doi.org/10.1162/neco.1995.7.1.108. URL: https://doi.org/10.1162/neco.1995.7.1.108.

[69] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[70] T. Akenine-Mller, E. Haines, and N. Hoffman. *Real-Time Rendering, Fourth Edition*. 4th. USA: A. K. Peters, Ltd., 2018. ISBN: 0134997832.

[71] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. USA: Prentice-Hall, Inc., 1996. ISBN: 0133737624.

[72] M. Pharr and R. Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005. ISBN: 0321335597.

[73] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, 2018. ISBN: 9780262536578. URL: https://books.google.de/books?id=7r34DwAAQBAJ.

[74] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CVPR* (2017).

[75] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *CoRR* abs/1701.04128 (2017). arXiv: 1701.04128. URL: http://arxiv.org/abs/1701.04128.

[76] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: (2016). arXiv: 1603.08155 [cs.CV].

[77] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (2015). arXiv: 1409.1556 [cs.CV].

[78] C. Wu, D. Bradley, P. Garrido, M. Zollhöfer, C. Theobalt, M. Gross, and T. Beeler. "Model-Based Teeth Reconstruction". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016).

[79] M. Jorgensen. "Iteratively Reweighted Least Squares". In: Sept. 2006. ISBN: 9780470057339. DOI: 10.1002/9780470057339.vai022.

[80] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980 [cs.LG].

[81] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic Differentiation in PyTorch". In: (2017). URL: https://openreview.net/forum?id=BJJsrmfCZ.

[82] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[83] O. Fried, A. Tewari, M. Zollhöfer, A. Finkelstein, E. Shechtman, D. B. Goldman, K. Genova, Z. Jin, C. Theobalt, and M. Agrawala. "Text-Based Editing of Talking-Head Video". In: *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3323028. URL: https://doi.org/10.1145/3306346.3323028.

[84] G. Zoss, D. Bradley, P. Bérard, and T. Beeler. "An Empirical Rig for Jaw Animation". In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201382. URL: https://doi.org/10.1145/3197517.3201382.

[85] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. "A Comprehensive Survey on Transfer Learning". In: (2020). arXiv: 1911.02685 [cs.LG].

[86] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng. "Learned Initializations for Optimizing Coordinate-Based Neural Representations". In: (2020). arXiv: 2012.02189 [cs.CV].