FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Abdurrehman Syed

From Words to 3D Faces: A Real-Time Pipeline for Avatar Expression Animation

Master's Thesis
Degree Programme in Computer Science and Engineering
May 2025

Syed A. (2025) From Words to 3D Faces: A Real-Time Pipeline for Avatar Expression Animation. University of Oulu, Degree Programme in Computer Science and Engineering, 56 p.

ABSTRACT

Generating realistic and controllable 3D facial animations from textual descriptions remains a challenging task, often requiring complex mappings from text embeddings to the latent space of 3D Morphable Models. This thesis demonstrates the possibility of integrating state-of-the-art image diffusion models and 3D face reconstruction for generating text-based animatable avatars and the practical challenges of doing so. My approach takes a neutral 2D facial image and a text prompt describing a desired expression as input. These are fed into a fine-tuned text-to-image inpainting model (based on DiffEdit with LoRA on the KDEF dataset) to generate a 2D image of the face exhibiting the target expression. To ensure texture consistency, a skin tone matching technique is applied to the generated image with respect to the original neutral face. Subsequently, both the original and the generated 2D images are processed using a 3D reconstruction network (DECA) to obtain textured 3D meshes. For enhancing visual realism, a post-processing phase refines the reconstructed 3D model features. Finally, continuous animation is achieved by interpolating among the 3D meshes that represent different textual input, enabling real-time text-based control over 3D facial expressions, demonstrated in a Unity context. This hybrid 2D-3D approach presents a practical and potentially more natural method for generating dynamic 3D facial animation from text input, with potential application to a range of applications in virtual reality, character animation, and human-computer interaction.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

3D	Three-Dimensional
2D	Two-Dimensional
HCI	Human-Computer Interaction
VR	Virtual Reality
AR	Augmented Reality
GAN	Generative Adversarial Network
DECA	Detailed Expression Capture and Animation
3DMM	3D Morphable Model
NeRF	Neural Radiance Fields
LoRA	Low-Rank Adaptation
LPIPS	Learned Perceptual Image Patch Similarity
SSIM	Structural Similarity Index Measure
PSNR	Peak Signal-to-Noise Ratio
OBJ	Wavefront Object File Format
CLIP	Contrastive Language–Image Pretraining
StyleGAN	Style-based Generative Adversarial Network
TediGAN	Text-guided Diverse Image Generation via GAN
FET	Facial Expression and Text
SIGGRAPH	Special Interest Group on Computer Graphics and Interactive Techniques
InstructPix2Pix	Instruction-based Pix2Pix
FLAME	Faces Learned with an Articulated Model and Expressions
EMOCA	Emotional Capture and Animation
MoFaNeRF	Morphable Face NeRF
PIRenderer	Portrait Image Renderer
FDNeRF	Few-shot Dynamic Neural Radiance Fields
Face2Faceρ	Real-time Face2Face with refinement
ArcFace	Additive Angular Margin Loss for Deep Face Recognition
IPGAN	Identity Preserving GAN
DiffusionRig	Diffusion-based Facial Rigging
InstaFace	Instant Identity-Preserving Face Editing
IP-FaceDiff	Identity-Preserving Face Diffusion
DiscoFaceGAN	Disentangled Controllable GAN for Identity-Preserving Face Generation
SIGGRAPH Asia	SIGGRAPH Asia Conference
ECCV	European Conference on Computer Vision
ICCV	International Conference on Computer Vision
DiffEdit	Diffusion-based Semantic Image Editing with Edit Mask Generation
U-Net	U-shaped Convolutional Neural Network
MSE	Mean Squared Error
KDEF	Karolinska Directed Emotional Faces
DDIM	Denoising Diffusion Implicit Models
JSON	JavaScript Object Notation

GPU	Graphics Processing Unit
GUI	Graphical User Interface
PNG	Portable Network Graphics
UV	Texture Coordinate Mapping (U = horizontal axis, V = vertical axis)
Lab	Lightness and Color-Opponent Dimensions Color Space
SSH	Secure Shell
I/O	Input/Output
RPC	Remote Procedure Call
HPC	High-Performance Computing
CPU	Central Processing Unit
FPS	Frames Per Second
rsync	Remote Synchronization Utility
Viseme	Visual representation of phonemes (used in lip-sync and speech animation)

1. INTRODUCTION

Facial animation and natural 3D avatars are increasingly becoming a primary research interest in computer vision, computer graphics, and human-computer interaction (HCI). Natural avatars are the foundation of applications such as virtual reality (VR), augmented reality (AR), video games, digital assistants, and interactive environments. These avatars must convey an enormous range of emotions, facial expressions, and subtle interactions in real time, which requires models that can produce not just visual appearance but also the dynamic, animated facial expressions of these kinds of characters.

Synthesizing realistic 3D facial avatars from photos has been an open question in computer vision. Earlier approaches typically required a number of input images or videos to generate 3D models with some level of accuracy, at times requiring costly hardware in the form of motion capture systems or an array of cameras. Recent breakthroughs have improved over the past few years in single-image 3D face reconstruction. A model like 3D Morphable Models (3DMMs) [1] has been shown to be capable of extracting a subject's identity and expression from a single 2D image. They model faces in terms of a set of parameters representing the geometry and shape variation of human faces, supporting realistic model and animation of facial expression from given blendshapes.

The introduction of generative models such as Generative Adversarial Networks (GANs) [2] and diffusion models [3] has further enabled one to manipulate faces at an extremely high level of realism. These can be trained to generate or modify faces based on some features such as age, gender, or emotion. While these models have demonstrated amazing potential in generating realistic 2D faces, extending their application to 3D face modeling and animation remains an ongoing focus of research. In particular, generating 3D facial expressions directly from 2D images in real time presents formidable difficulties, such as maintaining identity consistency while altering facial features according to dynamic expressions.

Another recent development is text-to-image synthesis in which natural language text descriptions can be synthesized into realistic images in models such as Stable Diffusion [4] and DALL·E [5]. These models are able to comprehend text input and construct images to represent the description; these models are trained on large data sets that consist of images and their text descriptions. Although these models have achieved impressive results on generic images, their application to the editing and animation of a real time human face has not been well explored.

Synthesizing 2D image creation with 3D reconstruction in this context offers a potential solution for generating realistic facial animations based on simple text instructions. By applying a 2D-to-3D pipeline, it is possible to get pre-trained models such as diffusion-based inpainting models and 3D face reconstruction networks to perform and generate extremely detailed, animated 3D faces from very minimal input (one photo and a text prompt). Being able to generate and animate 3D facial expressions in real-time would transform interactive applications that require lifelike, dynamic avatars.

The rest of this thesis is organized as follows. Chapter 2 presents the background literature and related work in text-to-image synthesis, 3D face reconstruction, and avatar animation. Chapter 3 presents an overview of the methodology, explaining the

proposed pipeline in depth, including expression editing, skin tone normalization, 3D reconstruction, texture enhancement, and animation. Chapter 4 describes integration of the system and deployment of a real-time distributed setup using multiple GPUs and client machines. Chapter 5 includes experiments and results, with a qualitative and quantitative evaluation for each module. Chapter 6 finishes the thesis with a discussion of limitations and issues encountered during development, future work, and an overview.

1.1. Motivation

While 3D face models and text-to-image models are powerful on their own, a wide gap exists for employing them together for 3D facial animation from text. Current methods for generating 3D facial expressions typically fall into two extremes. On one side, artists or algorithms manipulate 3D face rigs by hand or with performance capture, which provides fine control but is labor-intensive and not accessible to non-experts. On the other hand, end-to-end approaches like neural reenactment require a driving video to emit motion dynamics, which limits their application when such data is missing. What is lacking is a high-level, intuitive control modality for 3D facial expression generation. Natural language is an appealing choice: it allows anyone to specify a desired expression (or emotion) in words, which is far easier than fine-tuning several dozen blendshape sliders or taking a reference performance. Despite the recent breakthroughs in text-to-image generation, text-to-3D for faces has not received much attention. General image-to-3D synthesis methods (like DreamFusion [6]) are now available that optimize a 3D model (like a NeRF [7] or mesh) to best fit images rendered by a diffusion model. These, while available, employ iterative optimization that can take minutes or more per prompt and are applied to static scenes or objects, not animatable faces. In other words, existing text-to-3D pipelines are far from real-time and often do not produce riggable models suitable for animation. The motivation for this thesis is to bridge the above gap by creating a real-time pipeline for text-guided 3D facial expression animation. This kind of system would allow easy animation of digital faces. For example, in virtual reality and gaming, developers could quickly generate facial expressions for avatars by typing descriptions, speeding up content creation. In conversational agents or telepresence avatars, the agent's face could automatically animate to reflect emotional nuances described in dialogue or script. Importantly, by using a single neutral 2D photo as input, I personalize the 3D face to a specific individual, and then drive that face with different expressions using text. This avoids the need for extensive 3D scanning or multiple expressions of the person; a single photograph (which could even be the user's selfie or a character's portrait) suffices to create a fully animatable 3D face model. Real-time performance is a key motivation because it enables interactive applications: if one can generate and present each expression update in under seconds or even a second, a user can continually refine the prompt or utilize the system live in an application. Real-time visuals with high visual fidelity are hard to accomplish but new effective neural models (like latent diffusion and fast 3D mesh regression networks) make this increasingly feasible. Briefly, the goal is to develop a new framework that can support high-level, intuitive, and efficient control of facial animations, which is an urgent demand for computer graphics and

vision. By integrating text-to-image diffusion and 3D reconstruction, I envision a new era of text-to-3D avatar animation that is available and effective.

1.2. Problem Statement

This thesis addresses the problem of automatically generating a 3D facial expression animation from a single 2D image and a text prompt. Given one input image of a person's face in a neutral expression, and a textual description of a target expression (for example, "a person smiling with eyes widened"), the goal is to produce a smooth animation that transforms the neutral face to the target expression in 3D. Several core challenges are inherent in this problem:

- **Identity Preservation:** The generated expression must appear on the same person as in the input image. All identity-defining features (face shape, skin tone, and appearance details) should remain consistent through the animation, changing only what is necessary for the new expression. This is critical because facial editing should not alter who the person is, only how they express emotions.
- **Expression Realism:** The target facial expression should be convincing and true to the prompt description. The nuances of the expression (such as the curvature of a smile or the raise of eyebrows in surprise) need to be accurately reflected. The method should handle a variety of expressions (happiness, anger, surprise, etc.) and produce natural-looking results without distortions.
- **Smooth Interpolation (Animation):** Unlike having a single static "before and after" image, the problem is to create an animation path. That is, it is to calculate in-between frames or 3D states between neutral and maximum target expression so that the transition is realistic and smooth. The strength of the expression should increase or decrease smoothly without abrupt transition. Essentially, the solution must interpolate between two facial states in a visually continuous manner.
- **Text-Driven Guidance:** The only input specifying the desired expression is a piece of text. The system must correctly interpret the prompt (which might be abstract, e.g. "a delighted surprise") into the corresponding facial changes. This needs a semantic understanding of language in terms of facial feelings. The challenge is to ensure that the text guidance equates to visual change correctly, and solely the change intended.
- **Real-Time Performance:** The entire pipeline, from reading the input image and prompt to outputting an animated 3D face—should be efficient enough for practical use. "Real-time" in this context implies that a user would not have to wait long (on the order of a few seconds or less for the whole animation to be prepared). This constraint rules out extremely slow optimization-based techniques and demands a streamlined approach possibly using pre-trained models and fast algorithms.

In brief, this thesis is faced with the challenge: How can I take a single neutral face image and a text description of an expression, and quickly produce a realistic 3D face animation that brings the described expression to life on that person's face? Solving this problem requires advances in integrating text understanding, image generation, and 3D reconstruction/animation in a unified system.

1.3. Research Objectives

In order to tackle the above problem, this research sets the following objectives:

- **Develop a Text-Guided 2D Facial Expression Editing Module:** Adapt and fine-tune diffusion-based image generation models to modify a neutral facial image according to a text prompt. This will involve generating a photorealistic face image with the desired expression, retaining very little identity details. Techniques like DiffEdit [8] will be employed to recognize automatically the region of edit (e.g. lower face for smiling) and retain the background and non-expression areas as identical to the input.

- **Ensure Visual Consistency via Color and Detail Transfer:** Apply a mechanism to maintain consistent skin tone and brightness across the original image and the edited image. For instance, apply histogram-based skin tone matching for color distribution adjustment of the inpainted regions such that the edited 2D face becomes seamless. This operation will prevent visual jumps or artifacts in case later the given neutral and edited images are used for 3D reconstruction.

- **Reconstruct Detailed 3D Face Models from Images:** Use a state-of-the-art monocular 3D face reconstruction network (DECA [9] or equivalent) to reconstruct both the original neutral image and target-expression edited image as 3D face meshes. Both reconstructions must deliver textured 3D meshes with the subject's identity. Particular attention will be placed on expression geometry fidelity and texture detail levels (wrinkles, etc.) on the target-expression mesh, possibly by leveraging DECA's UV displacement map output. This objective ensures that I obtain two 3D models (neutral and expressed) of the same person.

- **Refine and Blend Textures for Realism:** Apply post-reconstruction texture refinement so that high-frequency details from the edited 2D image (e.g. teeth visibility in an open smile, or crinkling around the eyes) are properly transferred onto the target 3D mesh's texture. If the direct output from the reconstruction is blurry or missing details, enhance it by projecting the edited image onto the mesh or blending it with the original texture. The result should be a high-quality, consistent texture map for the face in both expressions.

- **Implement Expression Interpolation and Animation:** Design a method to interpolate between the neutral and target expression in the 3D domain. This may involve interpolating the 3DMM expression parameters or performing vertex blending between the two meshes over time. The objective is to generate a

sequence of intermediate 3D face models that gradually morph from the neutral expression to the target expression, creating a smooth animation. The output should be a set of animation frames (or an animation clip) that can be rendered or used in real-time engines.

- **Achieve Interactive Frame Rates or Short Latency:** Optimize the pipeline so that each step (editing, reconstruction, etc.) is as fast as possible. This could include using optimized neural network inference (e.g. running the diffusion model at lower resolution then upsampling, or using GPU acceleration for DECA), and possibly precomputing or reusing certain computations. The goal is for the entire process from input to animated output to run in near real-time (at best, a few frames per second or faster, allowing near-instantaneous feedback).

  Through these objectives, the research will create a complete pipeline that takes textual commands and outputs animated 3D faces, while meeting requirements of realism and speed.

1.4. Author's Contributions and the Role of Artificial Intelligence

- **Writing:** The initial draft of the thesis was created by me. Afterwards, I utilized AI tools (OpenAI's ChatGPT and Google Gemini) in assisting with formatting structure and refining clarity. Additionally, helping with heading ideas and presenting complex topics in simpler terms. However, all content generated by AI has been thoroughly reviewed and revised.

- **Research:** The entire algorithm and pipeline design as well as the implementation of the code has been done by me. However, AI assisted as a search engine for finding and understanding the state of the art (related works) and also helped with debugging errors in my code.

2. RELATED WORK

This section explores prior work relevant to my goal of real-time text-to-image facial expression editing of 3D avatars from a single image, with identity preservation. I will briefly describe significant advances in the three general areas of text-to-image facial expression editing (Section 2.1), 3D face reconstruction from 2D images (Section 2.2), and facial animation and expression interpolation techniques (Section 2.3). Next, I will discuss methods for preserving identity in generative models (Section 2.4). Having seen the strengths and weaknesses of these current approaches will put me in a good position to present my suggested method and its contributions in the coming sections.

2.1. Text-To-Image Facial Expression Editing

Recent studies have examined controlling or editing facial expressions in images using text instructions, generally with the help of powerful vision-language models like CLIP [10]. StyleCLIP [11] launched text-controlled face editing by cross-bridging a pretrained StyleGAN generator and CLIP's semantic knowledge. StyleCLIP offers different methods (latent optimization, mapper networks, and global directions) for applying text-controlled edits (e.g., "make face smiling") to an input face image. One of the strengths lies in its attribute flexibility; StyleCLIP is able to control expressions (smile, frown, etc.) without explicitly having to have expression labels by using descriptive prompts. Identity drift can still occur for large edits, since the StyleGAN latent space can get identity and expression mixed up. The authors note that previous text-based models like TediGAN [12] that projected image and text into a common latent space and performed style-mixing did not accomplish the same semantic alignment – StyleCLIP's changes are better reflective of the desired text semantics. One of StyleCLIP's limitations is its computational cost: per-edit optimization takes a long time (minutes) and it does not generate 3D-consistent edits directly (only working on 2D images).

Fu et al. address certain of the aforementioned concerns in FaceCLIP [13], a text-to-expression synthesis GAN-based pipeline. FaceCLIP uses a multi-stage CLIP-guided generative model and introduces a new FET dataset of face images and text descriptions. FaceCLIP separates face attributes and excels in high-resolution photo-realistic generation of faces. The strong multimodal priors enable diverse expression editing ("angry face", "surprised face") with better identity preservation than one-stage models. Still, as a 2D method alone, FaceCLIP does not produce an animatable 3D face, it alters one image at a time.

Beside GANs, diffusion models have been used more recently in text-controlled face editing. InstructPix2Pix [14] is a general image editor that trains a diffusion model by fine-tuning it on edited and generated image-text pairs. It can execute commands like "make the person cry" by generating the edited image directly in a forward pass. Although very flexible, such diffusion editors may alter identity or other details if the text guidance invites massive changes, since they lack a face-specific prior. They also function only in 2D and do not have 3D consistency understanding. There are other works that incorporate 3D guidance. ClipFace [15] spans between text directions and a 3D Morphable Model (3DMM) [1] of the face. It creates both face

geometry and texture with a generative model, and as such, text not only can retexture or recolor a face but also alter its expression parameters. For example, from a neutral 3D face, ClipFace is able to create a smile or frown in both the mesh and texture via a CLIP-controlled latent manipulation. This produces 3D-consistent cuts and preserves identity in the sense of operating within the domain of a fixed 3DMM (the individual's identity is stored in the shape parameters). The limitation of ClipFace is dependence on 3DMM expressiveness – expressions that are extreme or minor beyond the basis of the 3DMM might not be covered. In the same spirit, FaceCLIPNeRF [16] utilizes a neural radiance field representation: it renders a customized 3D head from input images and then uses text-driven deformations to the NeRF [7] to alter expressions. The technique can render edited faces from new views and even generate depth (geometry) for the edits. It is also of the same high fidelity as NeRF but NeRF editing can be too time-consuming and must be optimized per scene; real-time performance is thus a problem. Briefly, text-to-image facial expression editing has progressed from early GAN-based to high-fidelity multimodal and even 3D-aware solutions. My thesis will extend these works by employing text prompts to induce facial expression changes, but with a difference from previous 2D editing techniques: it will be coupled with 3D animation and must both be performed in real time and preserve identity.

2.2. 3D Face Reconstruction from 2D Images

Reconstructing a manipulable 3D face model from a single 2D image is an essential process for my work. A seminal paper in this space is DECA [9], which learns to regress a 3D shape of a face with detailed animatable geometry from in-the-wild photos. DECA outputs a coarse parametric face (based on the FLAME 3DMM [17]) along with a per-vertex displacement map to encode person-specific details such as wrinkles. Interestingly, it introduces a detail consistency loss to differentiate expression-created detail (e.g. smile lines) from identity-unique detail so that natural wrinkles that depend on expression are attainable. This made the face reconstruction fully animatable – one can change the parameters of expression and the fine details (wrinkles) move accordingly, which is not supported by previous models. But DECA's pipeline is offline and uses a fixed-topology mesh, so while it is great at animation, it isn't in a vacuum predicting real-time inference or generalizing to extreme expressions that are outside of the training data.

Building on DECA, EMOCA [18] specifically focused on the recovery of emotional expressions from a single image. EMOCA adds an emotion consistency loss during training such that the reconstructed 3D expression is consistent with the emotional content of the input image. That is, it uses a deep emotion recognition model to help drive the 3D face estimator. As a result, EMOCA's shape reconstruction is as good as previous methods but much better expression fidelity – the reconstructed faces show subtle or strong expressions correctly like in the original image. This concerns my thesis especially: if one plans to animate a face with expressions of, say, anger or joy according to text, then the initial 3D model must be able to show those expressions realistically. EMOCA demonstrates how adding perceptual objectives (emotion, in this case) improves expression quality, though it still operates within the 3DMM

limits (which may not capture all nuances of, for example, muscle tension in extreme expressions).

A few of the more recent methods attempt even more realistic or expressive 3D faces by bridging parametric models and neural implicit representations. Riggable 3D Face Reconstruction [19] presented an end-to-end network that jointly estimates a personalized face rig (identity + blendshape basis) and per-image expression parameters, using a differentiable in-network optimization to impose physical constraints. By approximating a subject-specific rig from a single input, such an approach goes beyond a static mesh – it produces a riggable model that can be driven by new expression coefficients. The advantage is that subsequently, interpolating or retargeting expressions can be achieved via the recovered rig. Bai et al. show that this leads to state-of-the-art reconstruction accuracy and generalization robustness, enabling downstream animation applications like video-driven reenactment. In my instance, this kind of rig may be supported by text-described expressions. The drawback is that the optimization makes it heavier than regression-only methods but achieves a good compromise between accuracy and controllability.

Another area of work applies Neural Radiance Fields (NeRF) and hybrid models for facial reconstruction. MoFaNeRF [20] is a parametric NeRF model that scales a single image to generate a 3D model which can render photorealistic views as well as be morphed in expression and appearance. It addresses the identity, expression, and appearance as distinct codes inputting into a NeRF, taking advantage of the strengths of 3DMM (controllable parameters) and realistic rendering of NeRF. MoFaNeRF is therefore able to generate a high-fidelity face, rotate it (novel view synthesis), and even change expression by interpolating the expression code. The strength in this case is realism; hair, eyes, etc., are implicitly represented, therefore more realistic-looking than the textured mesh, and smooth expression change is achievable by code interpolation. NeRF-based solutions tend to need a couple of images or several views to train; MoFaNeRF covers single-image fitting but at the expense of aggressive optimization. Real-time animation of a single image NeRF is still problematic because of the rendering speed.

Latest, generative models have been used to generate fully animatable 3D avatars from a single photo. Chen et al. [21] introduce a Morphable Diffusion model: they fine-tune a 3D-aware diffusion model guided by a 3DMM to enable it to generate novel views and novel expressions of a person from a single input image. In their pipeline, once a human face is reconstructed, the diffusion model is able to render the face of that person with a new expression (and other viewing angle) yet still photorealistic and identifiable. Crucially, they demonstrate for the very first time a single-image process generating a photorealistic animatable head of an unseen individual. This is achieved by conditioning the diffusion on the 3DMM parameters (expression, pose) and training it on multi-view facial data. The result is remarkable fidelity but diffusion inference is relatively slow and not trivial to do in real-time. One also needs to provide it with expression parameters (which in my case would be extracted from text).

Briefly, I have robust solutions for converting one 2D image to a 3D face model with the ability to animate it. From my thesis' viewpoint, the DECA/EMOCA line offers regression directly as one means of gaining an animatable mesh with detail, and NeRF/diffusion techniques give even greater realism at higher computational cost. This thesis will most likely utilize a light-weight 3DMM-based or hybrid method to

facilitate real-time animation, taking advantage of these fragments to offer an aggregate base to ensure the reconstructed face has high-fidelity expression capabilities.

2.3. Facial Animation and Expression Interpolation

With a manipulable 3D face, the next task is to animate it according to target expressions. Classical facial animation will interpolate expression blendshapes or 3DMM parameters. Recent learned facial animation research has covered driving faces from other modalities (video, audio) or one-shot reenactment, but these are directly related to expression interpolation and can inform our text-driven solution.

One line of work is one-shot face reenactment, where a single source image is animated to follow the expressions of a driver (e.g., another face or some reference). Warping-based methods are common here: MarioNETte [22] introduced an image attention and landmark transformer technique to warp a source face according to a target's landmarks while preserving the source identity. By disentangling identity geometry and expression geometry, their landmark transformer allowed expression transfer without identity distortion. MarioNETte demonstrated high-quality reenactments even when source and driver have very different identities, a scenario that often causes identity leakage. This is relevant to expression interpolation in that their method can smoothly apply the driver's facial movements to the source face. However, MarioNETte still requires an example driver sequence; it doesn't on its own generate an expression trajectory. In my case, the "driver" will be a text prompt, so I need a way to convert text to a sequence of expression parameters – essentially a trajectory through expression space, which is a different challenge.

Another approach is to leverage parametric motion models. PIRenderer [23] directly employs 3DMM parameters to drive a Portrait Neural Renderer for expression and pose. Any expression coefficients (and pose) can be fed into PIRenderer, and the model renders a photorealistic face with those motions. This system shows one of the benefits of 3DMM-driven animation: by adjusting the expression coefficients as a function of time (e.g., interpolating between neutral and smile), it is easy to achieve a smooth animation. The model makes the output look real and just like the source person. For example, PIRenderer can render an input face and animate it directly by feeding in a time series of expression vectors (perhaps from another video or pre-defined curve). A limitation is that it trains on videos of talking faces and is primarily designed for reenactment or pose retargeting; generating expressions from scratch (especially from text) was not its focus. PIRenderer's approach in the thesis suggests using a neural renderer conditioned on expression parameters for real-time animation. But the next question is then how to get those parameters out of text – something that I will have to likely address through an intermediate mapping (e.g., text to expression vector).

Some works also combine warping and synthesis to enhance performance. Face2Faceρ [24] is a high-speed and high-fidelity one-shot reenactment system. It is a hybrid approach with 3DMM assistance that warps the source image according to the expression of the driver (with 3DMM landmarks) and then completes filling in the details with a lightweight network. Using 3DMM, Face2Faceρ renders identity, expression, and pose (these parameters are provided by the 3D model) and thus eliminates most of the artifacts. Real-time performance is where the power lies,

and interactive animation is where this matters. The technique shows that using a parametric model (3DMM) not only improves quality but also performance since the majority of the work (predicting large motions) is done through warping. For my thesis' real-time output requirement, this is good news – I can use an equivalent idea of applying parametric warps (for expression change) and only a small network for details, and get fast frame rates. But Face2Faceρ still has an explicit driver per frame (video-based reenactment). If I were to use it on text, then I would have to create a sequence of expression parameters from the text prompt (e.g., text is saying "slowly go from neutral to huge smile" – I would create that parameter path).

Another idea that one can think of is few-shot dynamic avatar models. FDNeRF [25] learns a dynamic neural radiance field from a face with few images and enables expression editing and reenactment on the identity. It presents a conditional feature warping module that registers multi-view features of diverse expressions and makes it possible for the NeRF to render the face with unseen expressions not included in the inputs. FDNeRF can be taken as input a source face and, for example, an expression code, and produce a new view of the face making that expression, even extrapolating beyond the expressions in the training images. Its output includes smooth expression interpolation between them and even video-driven reenactment for any new sequence of expressions. The approach surpasses previous dynamic NeRFs on both reconstruction and expression editing. While FDNeRF generates high quality, NeRF-based rendering is often slower than real-time (though research continues on real-time NeRF). In addition, it requires a few input images (not single-shot). In the context of my thesis, FDNeRF's idea of an identity-adaptive neural face that can be re-posed and re-expressed is motivating – essentially, it provides the technique of obtaining unlimited views and expressions of a person from limited data. If real-time rendering is possible (perhaps with a quickened neural renderer or even a mesh conversion), then this kind of model would be text-driven. For now, I could otherwise utilize a mesh method for its speed, but FDNeRF's success indicates that high-quality expression interpolation can be done even on very sparse input.

Finally, I will mention works that focus on good identity preservation across animation (which overlaps in part with the next section also). The recent work is by Oorloff & Yacoob [26] where they propose a StyleGAN latent-based reenactment method. They decompose a source image into an identity latent and a facial deformation latent in the StyleGAN space and then recombine these with the motion of a driving video to generate reenacted frames. By a cycle-based updating, they get high-definition (1024^2) reenactments that are robust against the source's original expression and pose, i.e., even when the source face would have been smiling, it can be reenacted with a fresh expression without loss of identity. This overcomes expression interpolation: their system can erase the source expression and replace with a new one from the driver. The important insight in this case is to separate expression from identity so that the subject can continue to be recognized across animations. In text-based animation, I will also want the text to control only the expression facet and not alter the identity (and other attributes like lighting). The downside of Oorloff's and similar latent approaches is that they rely on a well-pretrained generator (StyleGAN2) and an encoder, which can be advanced and may not provide 3D consistency (even though certain latent approaches produce very realistic 2D videos).

In general, facial animation has several approaches to choose from: warping-based (identity-constrained and efficient for subtle movements), GAN-based (high fidelity, latent-controllable), and NeRF/diffusion-based (3D-coherent, photorealistic but time-consuming). For the thesis, real-time text-to-expression animation is something that can be achieved by applying a parametric model (3DMM) to bridge the gap between text and expression. I can employ learning from one-shot reenactment: use a neutral expression from a single face, then interpolate expressions according to the text prompt, e.g., reenactment but with the "driver" generated. Smooth interpolation can be solved by mixture of expression coefficients or using a small network to smooth transitions (e.g., Face2Faceρ). The papers surveyed provide a set of techniques for achieving this, and my system will likely use some combination of them e.g., an expression-controlled face rig based on 3DMM (from Section 2), with expression parameters interpolated or synthesized from text descriptions, and maybe further augmented by a neural renderer for photorealism.

2.4. Identity Preservation in Generative Models

Preservation of the subject's identity in facial modifications or animation is a significant concern. Generative models will naturally warp identity when creating dramatic expression or pose variations, as identity attributes can get confused with those. Previous research has explicitly aimed for this by separating out identity representations or by adding identity similarity constraints on generation.

One influential work is DiscoFaceGAN [27], which introduced a 3D-guided GAN where the latent space is split into identity, expression, pose, and lighting variables. By training with imitative-contrastive learning using a 3DMM prior, DiscoFaceGAN is able to vary expression or pose while keeping the identity code fixed, thus maintaining the same identity in generated images. For example, it can generate a person smiling or frowning and the face still looks like the same person. This explicit disentanglement is highly relevant: my thesis aims to do exactly this in the context of animation. The strength of DiscoFaceGAN is its precise control and high-quality output; however, it was demonstrated on generating synthetic faces (not necessarily editing a given real face, unless one projects the real face into the latent space). Also, being a generative approach, it might not perfectly reconstruct the input identity unless inversion is very accurate. I can draw from this the importance of using a strong identity prior (like a face recognition feature or a 3DMM shape) to anchor the identity.

FaceShifter [28] initially was a face swap model, but it made a big leap forward by using an identity preservation network for maintaining the source identity in the face-swapped result. In the case of editing expressions, the same idea is to utilize a pretrained network of face recognition (e.g., ArcFace [29]) to encourage the edited face to embed into the same identity as the input. FaceShifter's results were much more identity-preserving than earlier swaps, demonstrating that using generative models with direct identity guidance can reverse identity drift. Most of the follow-up researches (e.g., SimSwap, IPGAN) followed this paradigm. For my project, while I did expression animation I can employ an identity loss between the original face and the animation frames in a way that the identity features remain close even when the expression is changing. One limitation to note is that overreliance on identity loss will

sometimes conflict with expression changes (e.g., a very large expression will subtly alter facial contours, and identity loss would seek to prevent this), so compromise must be found.

The other breakthrough is DiffusionRig [30], which applies a diffusion model to a person's face, learning personalized generative priors that strongly preserve identity. DiffusionRig is provided with 20 images of a person and learns to "rig" that face with controllable parameters (expressions, lighting, pose) via a conditional diffusion process. The result is identity-specific and can generate the subject under different expressions while being very loyal to his or her identity and high-frequency details. In experiments, DiffusionRig achieved state-of-the-art photorealism and identity retention in edited photos of the subject. This is essentially an overfit model by identity, which is not one-shot deployable (since it needs a great number of images to fine-tune).Yet it is an extreme where identity is almost entirely retained (as the model memorizes the person).My thesis configuration is one-shot and hence I cannot fine-tune a big model per person. But I can take inspiration: using the person's single photo to extract as much identity information as possible (e.g., through a 3D morphable model identity shape, or an identity latent in a generative model) and then not changing that during expression changes. Further, I can employ feature fusion techniques e.g., merging an identity embedding with an expression-conditioned generator – so that identity features are not overwritten by expression changes.

Very recent approaches specifically focus on identity preservation for various edits. InstaFace [31] is a facial appearance editor based on diffusion that offers a novel guidance for identity preservation from a single input image. They include multiple 3DMM-based conditionals and a facial recognition embedding in the generation process to maintain the same identity, background, and accessories in the edited output. The authors obtain higher identity preservation and photorealism than current state-of-the-art editors, even in extreme pose, expression, or lighting edits. This method is particularly well-suited as it is tuned for my goals and operates within a challenging one-shot setting. The use of 3DMM in this example provides a structured method of controlling geometry (so identity geometry can be set or guided), and the facial recognition capabilities ensure top-level identity features (e.g., the shape of eyes, nose, etc.) are maintained. One likely limitation is that extremely large expression changes will still cause small identity drifts (since, e.g., a very large laugh can inflate cheeks and close eyes, altering the structure of the face). But overall, InstaFace's architecture is a recipe for how to enact identity: include as much prior knowledge (3D shape, identity embeddings) as is feasible into the generation process.

In the video environment, IP-FaceDiff [32] targets text-directed facial video edits and preserves identity as the primary goal. Their approach fine-tunes a diffusion text-to-image image for videos and incorporates an explicit optimization of identity preservation at edit time along with temporal consistency constraints. The result is the ability to process text commands (e.g. "add a smile") on a video of a person and get back a new video with the person smiling, without changing who the person is in all frames. They also manage to process faster using pretrained models. For my thesis, which is perhaps not on long videos but short animation, the idea of fine-tuning or optimization with identity and temporal constraint is valuable. IP-FaceDiff basically confirms that with effective constraints, text edits can be applied to a sequence and still retain identity coherence frame by frame. One limitation in diffusion models

(that is relevant here) is potential frame differences (flicker), but loss of them over time addresses that. This thesis cannot, in a real-time system, necessarily allow a big diffusion model, but I can have identity loss and temporal smoothing as part of my model training.

Finally, notice that identity preservation is not merely a matter of network architecture but also of representation choice. The kind of papers like Disentangled Representation on megapixels [24] and others have used contrastive learning to set one latent to only affect identity. These ideas are in line with what I have discussed: keep an independent "ID code" which the model cannot just override as it changes expression. My thesis will likely employ a related idea, e.g., using the individual's 3D neutral face shape or its encoding as an identity carrier that remains fixed when expression changes. I can also employ an identity loss on the output renders (with respect to the input image's features) to regularly enforce that the individual "looks like themselves" in every frame of the animation.

2.5. Comparison and Thesis Contribution

The research in the surveyed papers informs us that all of text-based face editing, 3D face reconstruction, facial animation, and identity preservation have each made significant progresses individually. My thesis aims to integrate all these into one system: with a single 2D image of a person, I will create a 3D avatar and animate it in real-time with input from text, preserving the person's identity. This is an addition to the state of the art in several ways. Firstly, in contrast to earlier text-to-image editing methods, my system will generate a full sequence of expression changes (animation) and output it in 3D, enabling new views as well as more immersive applications (e.g. VR/AR or telepresence). Second, unlike typical 3D face reconstruction pipelines that just cut short at producing a static model, I extend the process to dynamic expression synthesis from high-level descriptions – essentially putting semantic animation on top of reconstruction. Third, unlike other facial reenactment and animation methods that require a driving video or audio, my system uses an intuitive modality (natural language) for driving the animation. This provides new opportunity (e.g., "make the avatar curious-looking, then laugh") that is hard to specify with a reference video. Then I go one step further in transcending identity preservation by introducing the latest concepts (disentangled representations, identity losses) into the text-based animation environment. The final outcome will be a new system that, for the first time, integrates text-guided expression control, one-shot 3D avatar creation, real-time animation, and identity maintenance. I expect the new method to generate more controllable and identity-maintaining facial animations compared to previous work. For example, where a baseline image-to-text editor would output a single smile image with little identity change, this system would output a smooth 3D smile animation of the actual person's face, retaining their principal features everywhere. In short, by standing on and merging the works described, the thesis will advance the state of the art towards truly interactive and personalized 3D facial expression animation.

3. IMPLEMENTATION

My real-time 3D facial expression animation pipeline is composed of several stages working together to transform a static neutral face image into a smoothly animated 3D face with varying expressions. In general, the system consists of: (1) a fine-tuned generative model that produces expression-specific image edits, (2) a real-time inference mechanism to apply text-prompted edits to a neutral face, (3) a post-processing step for skin color consistency, (4) a 3D reconstruction module to obtain a deformable face mesh from the 2D output, (5) a texture enhancement procedure to improve visual fidelity, (6) an interpolation scheme to generate intermediate frames between expressions, and (7) a Unity-based application for rendering and user interaction. These components are distributed across a computing cluster and a local workstation, where they run in parallel for performance optimization.

Figure 1 illustrates the system architecture and data flow in general, with image generation and reconstruction done by the cluster, and animation interpolation and rendering done on the local machine.

The pipeline operates as follows. First, I fine-tune a Stable Diffusion model to specialize in facial expression edits using a DiffEdit-based approach (Section 3.1 Fine-Tuning with DiffEdit [8] and LoRA [33]). This model, given a neutral face image and an expression prompt, can generate a new image where only the desired facial expression changes while other aspects remain the same. Next, a real-time inference script uses this model to produce expression edits on the fly in response to user-provided text prompts (Section 3.2 Real-Time Inference Pipeline). After each image is generated, I apply a color correction step to ensure the skin tone and lighting remain consistent with the original neutral frame (Section 3.3 Skin Tone Correction via Histogram Matching [34]). The corrected 2D result is then passed to a 3D face reconstruction module based on the DECA [9] framework, which yields a textured 3D mesh of the face with the new expression (Section 3.4 3D Face Reconstruction with DECA). I then refine the texture map through filtering techniques to enhance important facial details, and apply an interpolation algorithm that blends the geometry and textures of consecutive expression meshes to produce smooth transitions (Section 3.5 UV Texture Enhancement and Interpolation. Finally, the sequence of 3D meshes is loaded into a Unity engine environment for real-time visualization. The user interacts with the Unity application to trigger new expressions, which in turn feeds back into the generation pipeline, completing the loop. In addition to that, throughout this pipeline, I leverage parallel execution on multiple GPUs and synchronized file transfers between the cluster and the local machine to achieve near real-time performance (Section 3.6 System Integration and Real-Time Visualization). I discuss each of these components at length in the following subsections.

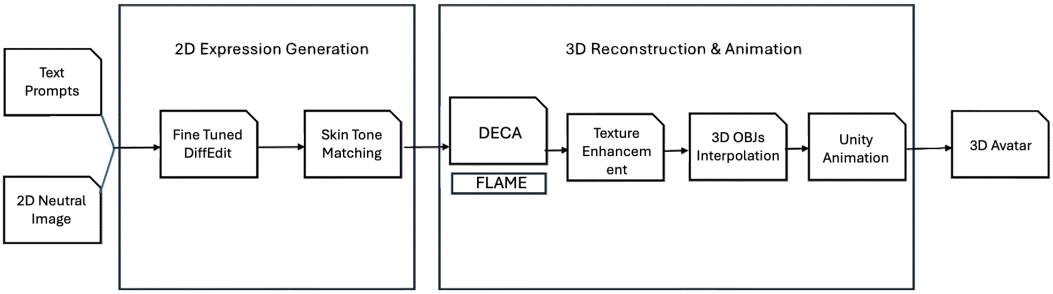

Figure 1. Proposed Model Architecture

Figure 1 provides a high-level system pipeline: the system takes a neutral face image and a text description as input, produces an edited 2D face with the target expression, reconstructs a 3D face with texture, and renders it in real time.

3.1. Fine-Tuning with DiffEdit and LoRA

To enable text-guided expression changes on a target face, I pretrained a latent diffusion model (Stable Diffusion v2.1 [35]) for face expression editing. Rather than training from scratch, I applied the DiffEdit framework with Low-Rank Adaptation (LoRA) to efficiently specialize the pre-trained model for my task. DiffEdit is a newly introduced technique for semantic image editing using diffusion models to automatically generate an edit mask via a comparison of model predictions made with and without the text prompt, thereby determining what regions of the image should be altered. This is done by editing only the facial features that belong to the new expression (i.e. the eyes and mouth for a smile), leaving all other facial features and background unchanged. By adding DiffEdit to my training, I was able to ensure that the model learns to apply its changes to the facial features relevant to each expression without requiring any labeling to the mask manually. In addition, DiffEdit employs a latent inversion step to preserve the original content, which helps maintain the identity and other high-level details of the input face even as the expression changes.

I fine-tuned the diffusion model using LoRA, which provides a lightweight way to adjust large models by injecting learnable low-rank matrices into the model's layers. For my use case, I froze the original Stable Diffusion weights and inserted LoRA layers of rank 16 in the attention blocks of the U-Net. This approach drastically reduces the number of parameters and GPU memory needed for training, while still achieving performance comparable to full fine-tuning. The LoRA module learns the "difference" needed to transform a neutral face into various expressions, and can be stored as a small adapter that is later applied on top of the base model. With LoRA, I would be able to fine-tune the model on a multi-GPU platform with little computational cost, which was crucial because of the high-res images and the need to iterate through many examples.

Dataset and Training Procedure: I curated a training dataset from the Karolinska Directed Emotional Faces (KDEF) [36] database, a widely used collection of face images with posed expressions. KDEF contains 70 individuals (35 female, 35 male),

each displaying 7 distinct emotions (neutral, happy, angry, afraid, disgusted, sad, surprised) captured from 5 camera angles. For the purposes of this thesis, I filtered the dataset to include only the front-facing (straight angle) photographs to avoid any confusing pose variation. This yielded approximately 980 images (70 subjects × 7 expressions × 2 sessions) of resolution 562×762 pixels, each subject having a neutral expression image and several emotional expression images. I paired each neutral expression image with one of that same subject's emotional expression images to form input and target training pairs. The neutral face served as the input (the image to edit), and the corresponding emotional face served as the target (the desired output). I also constructed a simple text prompt describing the target expression (for example, "happy face", "angry face", etc., using the emotion label from KDEF). This triplet (input image, target image, and text prompt) constituted a training sample for my diffusion-based editor.

Figure 2 shows various sample expressions from the KDEF dataset of the same person.

Figure 2. KDEF Dataset

This dataset was then used for fine-tuning the model. Fine-tuning was conducted on the CSC Puhti supercomputing [37] cluster using PyTorch and the HuggingFace Diffusers library. Two NVIDIA V100 GPUs were used in parallel: the first for training the diffusion model, and the second purely for concurrent processes (described in following sections). The training loop iterated over filtered KDEF pairs for a number of epochs. For each training example, the stable diffusion model (with 2.1 base weights) was conditioned on the input neutral face image and text prompt describing the target expression. I did not supply any manual mask; instead, DiffEdit's mask generation was employed on-the-fly. Specifically, the pipeline first performed a forward diffusion pass conditioned on an empty (or neutral) prompt to reconstruct the input image, and in

parallel a pass conditioned on the expression prompt. The difference between these two predictions determined the edit mask, essentially the regions of the face that change due to the prompt. This mask was then used to guide the model to only alter those regions when producing the final output, treating the task akin to masked inpainting but without a human-provided mask. I optimized the model parameters (specifically, the LoRA adapter weights) by minimizing a loss between the model's output and the ground-truth target image. The subsequent loss function used mean squared error (MSE) together with a perceptual similarity loss (LPIPS) [38] in an attempt to balance fine pixel-level correctness with high-level visual consistency. Masked areas were prioritized while calculating the loss such as setting so that expression-critical features were accurately restored while non-masked areas were pushed to be identical to the input. By doing so, the model could learn expression transformations that looked real and preserved the subject's identity and background.

Training was performed for a total of 50 epochs, with validation monitoring on a held-out set of image pairs to monitor performance. I observed that the training loss plateaued at around epoch 40, and the best validation scores (lowest LPIPS distance to targets) at epoch 50. Hence, I used the epoch-50 model to deploy. The learned LoRA weights at this point were extracted and saved as a standalone adapter file (tens of megabytes, a small fraction of the full model size). This LoRA adapter can be applied to the Stable Diffusion 2.1 base model to infuse it with the learned capability of expression editing. By keeping the base model fixed and modularizing the learned changes, I maintain flexibility – the base model's rich prior (learned from billions of images) remains intact, and only a narrow task-specific behavior (facial expression change) is added via LoRA. The fine-tuned model (base + LoRA) now forms the core of my expression generation system.

3.2. Real-Time Inference Pipeline

With the diffusion model optimized for facial expression edits, I created a real-time inference pipeline to use it repeatedly as the user inputs new expression prompts. The pipeline is encapsulated in a Python script run_diffEdit_lora_realtime.py, which orchestrates the loading of the model, listening for user input, executing DiffEdit-based edits, and saving the results for downstream processing. The script runs on the Puhti cluster (GPU 0) and interfaces with the rest of the system through a shared network file system.

When the inference service starts, it first loads the Stable Diffusion 2.1 model weights along with the LoRA adapter (epoch-50) to obtain the fine-tuned model ready for editing. It also loads a neutral reference image of the subject to be animated – this is the static neutral face from which all expression variations will be generated. To accurately preserve this specific face in all edits, I perform an image inversion step: The neutral image is encoded in the latent space of the model as an initial condition for diffusion. In practice, this involves a reverse diffusion such that the model can reconstruct the neutral image from the latent code. By starting from this inverted latent, the subsequent edits will better maintain identity and lighting, as the model "knows" the exact baseline to modify. This step corresponds to the latent inference component

of DiffEdit, ensuring the content outside the expression regions is preserved in the output.

After initialization, the script enters an event loop waiting for new expression prompts. Prompts are entered directly via the terminal in real time. Each prompt (e.g., "happy", "angry") is read and queued for processing. Upon receiving a new prompt, the system generates the corresponding expression image using the fine-tuned diffusion model.

When a prompt is received, the pipeline would initially construct a complete text description to condition the diffusion model. In simplicity, I use a simple template suitable for the fine-tuning context – i.e., if the prompt is "happy," the script construct a sentence like "a person with a happy facial expression" as conditioning text. This helps to provide the model with a clearer context (labeling "person" and "facial expression") and is similar to the type of captions the model may have seen during fine-tuning. I found that this templated approach yields more reliable results than applying the one-word prompt alone, as this reduces ambiguity.

Next, the neutral image's latent (from the inversion step) is fed into the DiffEdit pipeline along with the constructed prompt. The DiffEdit mechanism then generates an edit mask and the edited image in a two-step process (predicting with and without the prompt, as done in training). The guidance scale and diffusion steps are set such that the expression change is sufficiently strong to be recognizable, yet not so strong as to distort identity. Thanks to the earlier fine-tuning, the model requires relatively few diffusion steps to converge to a plausible output (I use 50 DDIM sampling steps for real-time efficiency). The result of this process is a new image in which the original neutral face now displays the requested expression. For instance, if the user prompted "angry face", the output image will have furrowed brows, tightened lips, and other characteristics of anger, while the person's identity, pose, hair, and background remain the same as in the neutral image. The automatically generated mask from DiffEdit confines changes largely to the upper and middle face regions in this case (brows and eyes, maybe slight mouth changes), preventing unwanted alterations elsewhere.

Once the generation is complete, the script writes the resulting expression image to the shared output directory. The filenames are date-stamped and ordered sequentially such that the new frame can be isolated. This output will trigger the rest of the pipeline steps: when the image file becomes visible, the 3D reconstruction step (in parallel run on another GPU) will detect it and begin processing (Section 3.4 3D Face Reconstruction with DECA). In summary, the real-time inference component continuously takes the latest neutral face (or last generated frame) and a text prompt to produce a new expression image, handing it off via the filesystem to downstream modules. Because the heavy computation (diffusion sampling) is done on the cluster GPU, the system can maintain interactive speeds — each image generation takes on the order of a few seconds, which is fast enough for a responsive animation cycle given the subsequent processing and rendering overheads.

3.3. Skin Tone Correction via Histogram Matching

While the DiffEdit-based generator produces expression changes of good quality, I observed minor differences in lighting and color tone between frames. In particular,

the skin tone of the face may subtly differ from the neutral image to the generated expression image (e.g., be slightly warmer or paler), due to the stochastic nature of diffusion sampling and the model's learned priors. Such subtle variations in color can introduce a flicker effect or an unnatural appearance when the images are used as textures on the 3D model over time or across different expressions. To alleviate this, I introduced a post-processing step to enforce color consistency, paying specific attention to skin tones.

After generating each expression image (but before 3D reconstruction), I carry out histogram matching in the Lab color space with the neutral face image as reference. I operate in the Lab color space because it separates luminance (L) from chromaticity (a and b channels), allowing us to manipulate colors in a perceptually uniform way without changing brightness.

The following approaches were taken:

- **Facial Region Isolation:** I first isolate the facial region in the generated image to target corrections. A simple approach is taken: I assume the face is roughly centered (as it is in the neutral/reference image and DiffEdit output) and apply a center crop or an ellipse mask covering the face region. In addition, I perform a basic skin color estimation by examining the reference neutral images' skin pixel values (in Lab space) and finding pixels within the target image with similar color values. These processes produce a mask that classifies the face (and skin) region in the generated image but not the background and non-skin regions like hair or clothing.

- **Histogram Matching:** I then adjust the color distribution of the new image's face to be identical to the neutral image's face. For each Lab channel, I compute the histogram of pixel values for both the neutral reference and for the new image (based on the face mask). I convert the new image distribution to match the reference. This will cause the mean and standard deviation of, say, the "a" (green–red) and "b" (blue–yellow) channels in the output to be the same as those of the neutral face. That is, if the diffusion model rendered the face slightly redder or brighter, this will be countered by drawing the histogram in the direction of the neutral face. The L channel (lightness) is treated gently so as not to change perceived shading too much; I primarily emphasize color uniformity, though gradual light variations are also possible through this matching.

- **Blend and Output:** The histogram-matched face region is then blended back into the image. I perform it carefully so that there are no sharp edges at the mask boundaries: a feathered mask or alpha blending creates a smooth blend between manipulated face pixels and unchanged background pixels. Pixels sufficiently similar in color to regular skin tones are altered, with background colors left unchanged. The impact is that the new image expression closely resembles the neutral image in coloration and overall brightness. The face appears to have been photographed under the same light as the neutral frame, improving temporal consistency when these images shall be utilized as textures later.

This color correction step is fast (a fraction of a second using OpenCV or similar libraries) and operates automatically on each generated frame. It significantly reduced

visible jumps in skin tone between consecutive expressions. The improved texture consistency makes the later interpolation (Section 3.5 UV Texture Enhancement and Interpolation) and rendering in Unity more convincing, as the user sees a stable skin appearance throughout the animation. In Figure 3, the effect of this correction can be seen – the post-processed image's skin hues more closely match the neutral reference, eliminating the slight pale cast that was present in the raw generated output. By incorporating this histogram matching stage, I ensure that my pipeline's visual output maintains a coherent look, as if all frames were captured in one continuous session with fixed camera settings.

Figure 3. Skin-tone Matching

3.4. 3D Face Reconstruction with DECA

Once a new expression image is generated and color-corrected, the task is to re-construct a 3D face model from this 2D image. For this I use DECA (Detailed Expression Capture and Animation), which is a state-of-the-art approach for monocular 3D face reconstruction. DECA is able to take a single input face image and predict a textured 3D face mesh, both the coarse shape (identity-dependent face structure) and fine-grained expression deformations. A key strength of DECA is its ability to disentangle identity and expression: it learns a parametric face model where the identity-specific geometry and expression-caused offsets are represented separately. This precisely matches this thesis' goal of one identity but with multiple expressions.

In my implementation, I run a custom script customDeca.py on the Puhti cluster (on GPU 1) which continuously checks the shared directory for incoming new images from the DiffEdit pipeline. When it finds a new expression frame (image file), this script automatically runs the DECA reconstruction for the image. The DECA model I use is pre-trained (from the authors' release) and predicts the 3D face in a parametric form (based on the FLAME model) along with albedo (texture map) and lighting parameters. I have slightly modified the standard DECA inference to enforce identity consistency

across all reconstructions. This is done by fixing the identity latent code to that of the neutral face:

- **Identity Calibration:** When the pipeline starts, I feed the neutral reference image (the same one used in the diffusion model) into DECA exactly once. DECA produces a set of latent parameters, including the shape code (which encodes identity-specific facial structure), the expression code, the pose, and so on, as well as a detailed albedo/texture map for the neutral face. I save the shape code (and optionally the high-frequency detail code and albedo) from this neutral reconstruction. This represents the person's intrinsic facial structure without any expression.

- **Expression-specific Reconstruction:** For each new expression image, instead of allowing DECA adjust all the parameters individually (which may lead to marginal changes in perceived identity or shape), I fix the shape parameters to the saved neutral identity code and keep them fixed. I then apply DECA's fit process to just the expression parameters, pose, and perhaps some marginal tweaks to detail/texture. Practically, this means that I will pass the neutral shape code to DECA as a constant so that the resultant 3D mesh of the new face has the same underlying structure (jawline, nose shape, etc.) of the neutral face, but with only expression changing. As DECA is designed to decouple these problems, this solution is effective – the model can still fit the new image by modifying the expression coefficients (which deform the mesh to smile, frown, etc.), but not the identity-defining coefficients. As set forth in the DECA formula, the person-specific identity information is not modified when I keep the person-specific latent constant, and only expression-dependent wrinkles and movements are added on top, so that natural animation of that identity can be facilitated.

- **Output Generation:** After fitting, DECA outputs a 3D face mesh in the form of a textured OBJ file. This includes the face geometry vertex positions (with around 40k vertices covering the face and head, depending on the FLAME model resolution used by DECA), along with a UV texture map image. The texture map is either captured from the input image (projecting the 2D photo onto the 3D model) or predicted by the network; in my case it manages to encode the face appearance as seen in the generated image. As I froze the identity, the output mesh shares the same vertex indexing and topology of the neutral face mesh, which is crucial for the subsequent interpolation. I save the OBJ and its equivalent texture (PNG image) to the shared output directory for use by later stages. Each expression image thus delivers an equivalent expressionXYZ.obj and expressionXYZ.png (texture) file.

The DECA reconstruction would typically take less than two seconds per image on a single GPU, and therefore it is feasible to process frames in real-time. By running this in parallel with the diffusion model (on a different GPU and process), I can hide its latency behind the generation of the next prompt. The result of this stage is a sequence of consistent 3D face models: each one has the same underlying neutral face shape and UV coordinates, with only the expressions and textures differing. This offers us a nice foundation to animate from, essentially converting the problem from image

interpolation to mesh animation, which is easier to handle for real-time rendering. Figure 4 illustrates an example of this output: the neutral input image is reconstructed as a neutral 3D mesh, and a generated "smiling" image is reconstructed as a mesh with the mouth and eyes dynamically altered to smile, but the rest of the face geometry is the same for both meshes.

Figure 4. DECA Output Generation

3.5. UV Texture Enhancement and Interpolation

Although the DECA algorithm provides me with a high-quality 3D mesh for each expression frame, the raw textures reconstructed by DECA may appear ever so slightly blurry or have poor details when rendered on the 3D model. This is partly because the input images (color-corrected or otherwise) could harbor some blurriness caused by the diffusion step, or DECA texture mapping could blur a few details to fit its statistical model. Moreover, inconsistency of texture sharpness across frames can appear when changing expressions. I solve these problems by performing a texture improvement pass on all UV texture maps after it has been rebuilt by DECA and streamed into the local machine.

The texture enhancement routine runs on the user's Mac workstation as part of the post-processing script which processes incoming OBJ files. I apply an unsharp masking technique to improve high-frequency detail in the facial texture and to denoise. That is, for each texture map (typically a 2D image of the unwrapped face, e.g., 1024×1024 pixels):

- I first apply a Gaussian blur to the texture image, using a small kernel (e.g., 5×5 or determined by the texture resolution) to create a blurred image. The blurred image keeps the low-frequency color distribution (overall skin tone, lighting gradients) but removes finer details like pores, wrinkles, or sharp edges around facial features.

- Next, I compute a high-frequency image by subtracting the original from the blurred one (i.e., high_freq = original − blurred). This high_freq image represents the detail that was smoothed out – including noise and genuine sharp features.

- I then amplify the high-frequency component by a certain factor (e.g., 1.5× or 2× amplification, empirically determined). This increases the contrast of fine details. However, to avoid amplifying sensor noise or compression artifacts, I clamp or threshold the high-frequency values to focus on relevant facial detail ranges (e.g., facial feature edges). In some cases, I amplify only the mid-high frequencies (to avoid amplifying single-pixel noise).

- Finally, I combine the enhanced high-frequency component with the blurred image (or even the original image) to obtain the improved texture. The result is a texture with smoothness derived from the Gaussian blur (removal of undesirable noise or DiffEdit artifacts) and clarity from the added details (eyes, eyebrows, lip borders, skin wrinkles etc. being more defined). In effect, this is a standard unsharp masking, which enhances the image by restoring an overdone version of detail lost in blurring.

I apply this filter to the face region of the UV texture, and if needed, I can leave out non-skin regions (like hair or background of the UV map) to avoid amplifying noise there—though in the UV map, typically only the face and ears are present. The enhanced texture is then saved, replacing or supplementing the original texture file associated with the OBJ. In cases where the enhancements might introduce slight color shifts or overshoot on edges, I fine-tune the Gaussian kernel size or strength per texture to maintain a natural appearance.

After this improvement phase, the textures visualized on the 3D model exhibit finer details such as more defined lip lines, better-defined eyes, and more realistic skin microstructure. These modifications are delicate yet crucial to procure realism, especially with the face revealed close-up in Unity. The operation is fast (tens of milliseconds per texture) and hence won't bring down the pipeline. By providing each frame's texture high-quality, I also help the stage of interpolation, as transitions between textures sequentially will be more of geometric change than blur differences or random noise.

Figure 5 shows the output of the previously used 3D DECA generated faces after performing the UV texture enhancement. Making the output, sharper, brighter, and more textured.

Figure 5. DECA Output UV Enhanced

Having obtained a sequence of discrete expression meshes (with corresponding textures) for each prompted expression change, the next challenge is to create a smooth animation when transitioning from one expression to another. If I simply cut between consecutive 3D static face meshes, the motion would appear unnatural and jerky. Therefore, I incorporated an interpolation module that generates intermediate frames between the key expression states. This interpolation is applied both in geometry space (vertex positions and normals) and in texture space, for a smooth transition from one face to another.

Interpolation Strategy: When two state of expressions A and B occur consecutively to each other in the timeline (for example, A might be "neutral" and B is "surprised", or B is whatever the user prompted next), I create an in-between short sequence of meshes that smoothly deform from the shape of A to the shape of B. Because every one of the expression blend meshes shares the same topology and vertex order (thanks to the same identity from DECA), I can interpolate them vertex by vertex. I choose an interpolation step number N based on the required smoothness of the animation (e.g., $N = 10$ in-between frames for a fairly quick change). For each step i between 1 and N, I compute an in-between vertex position for each vertex as:

$$V_i = V_A + \frac{i}{N}(V_B - V_A),$$

Which linearly interpolates shape A's coordinates to shape B. This linear interpolation can be well applied to small changes like between facial expressions since the change is well localized and the risk of self-intersection of the geometry is minimal. Then normals for each of the intermediate meshes are recomputed from the interpolated geometry so that they will be properly lit, although it is also possible to interpolate the normals themselves directly as a time-saving measure. I chose to recompute normals to prevent any shading artifacts with the blend.

Texture Blending: Simply interpolating the geometry will not be sufficient as facial texture (particularly things like wrinkles or the visibility of teeth when the mouth opens) also changes from one expression to another. I perform a matching interpolation between texture maps A and B. A naive linear cross-dissolve of the entire texture, however, can cause a ghosting effect where features from both expressions are semi-visible (such as half-transparent teeth or two eyebrows) in the in-between frames. I mitigate this by using a masked texture blending process:

I then create a difference mask between textures A and B. They can be done by computing their color space difference or by projecting the two textures onto a shared image (as both are UV-aligned) and finding areas of large change. For example, the mouth area would be very different if expression B has an open mouth (teeth showing) compared to A's closed one; also, eyebrow region differs if brows moved.

Based on this, I create a weighting mask that varies from 0 to 1 over the face, indicating which regions should favor texture B versus texture A at a given blend fraction. At the start of the transition, I want mostly A's texture; at the end, fully B's texture. But in the middle, for regions that change (like the mouth), a linear blend could look odd, so I instead try to gradually reveal new features. I can keep region close to the mouth with A's texture until a specific halfway mark when it switches to B's (in order not to draw half-transparent teeth). It is achieved by thresholding or easing the blend function in those regions. I essentially treat it as a morph cut where different areas of the face overlap with slightly different speeds in order to maintain a similar look.

For each intermediate frame i, the texture is thus computed as:

$$T_i(p) = (1 - w_i(p)) \cdot T_A(p) + w_i(p) \cdot T_B(p),$$

where p is a pixel/UV coordinate and $w_i(p)$ is a weight between 0 and 1. $w_i(p)$ is generally i/N (the global blend fraction) but modulated such that if p lies in a "changing region" (like mouth interior), w remains 0 for longer and then quickly transitions to 1. Outside those regions, w follows the linear fraction. This selective approach ensures that spectator does not see a faint double image of say eyes or mouth; instead those features snap or emerge smoothly at the appropriate time.

All the in-between texture images are then saved out, numbered in sequence, along with the corresponding OBJ files with interpolated geometry. The end result is a series of frames from expression A to B, which can be played as an animation. If the user triggers a series of expressions in sequence (e.g., neutral -> happy -> surprised -> neutral), I interpolate like this for each pair of consecutive keyframes. This basically produces an animation timeline comprising smooth morphs from one triggered expression to the next.

The interpolation script is executed on the Mac workstation right after a new key expression OBJ is received. It loads the old keyframe mesh (state A, say the current expression now being rendered) and the new incoming keyframe mesh (state B, just received) and subsequently computes the in-betweens as described. Default values: I generated 5–10 in-between frames per transition for demonstration, which at 30 FPS provides a transition animation of a few tenths of a second. This can be adjusted depending on how slow or how fast I want the change to appear.

By calculating these intermediate frames ahead of time rather than interpolating in Unity at runtime, I move the heavy lifting away from the game engine and offer smooth, artifact-free interpolation (since Python with numpy/OpenCV is capable of performing the image processing correctly). I also can inspect or modify the intermediate results if needed. The set of intermediate OBJs and textures are then made ready to be imported into Unity for viewing.

Figure 6 shows 5 frames created between the neutral and happy expressions using the linear interpolation script, these frames are later, used for smooth animation in unity.

Figure 6. Interpolation - from Neutral to Happy

3.6. System Integration and Real-Time Visualization

In this section, I discuss how all the components of the facial animation pipeline were connected together to build an end-to-end real-time system. This includes both the Unity-based visualization system and parallelized architecture for distributed computation.

The final step along the pipeline is the real-time display of the animated 3D face, rendered in a Unity program on a local machine. Unity acts as the rendering front-end, receiving 3D mesh outputs and textures generated by the processing pipeline and displaying them with smooth animation.

Unity Scene Setup: The Unity scene has a pre-built GameObject that gets animated as the face, with a Mesh Renderer and a default Standard Shader-based material attached to it. The object gets replaced dynamically with new meshes and textures that get generated by the pipeline. The Unity application monitors for new OBJ files and texture maps within a shared directory on the mac. An OBJ file is loaded, parsed for vertices, UVs, and faces, and the corresponding texture mapped to the material by a script customized to that purpose.

Animation Playback: The interpolation module in the pipeline generates sequences of OBJ files that encode the face morphing from expression A to expression B. The frames are loaded by Unity sequentially and rendered to create a smooth animation. The play process involves updating the Mesh Filter iteratively with each of the interpolated frames at a constant frame rate (e.g., 10 FPS), creating the appearance of continuous change of expression. This approach has seamless transitions with low overhead because all meshes share the same topology, and texture is blended with geometry changes.

Loading and Synchronization: Unity continues to monitor the common directory for newly added sequences of expressions. If it finds a new transition (i.e., from neutral to smiling), it will not halt the ongoing playback. Rather, the new interpolated frames are appended to the existing animations. In this manner, the avatar can continue to animate seamlessly as more and more expressions are added with the passage of time and accumulate a dynamic timeline of facial movements.

Performance Considerations: Since Unity's function is limited to loading precomputed assets and rendering, the performance is extremely smooth. Frame rates are easily above 60 FPS for rendering a single face, and the only noticeable lag is felt during the loading of new sequences. With computationally intensive tasks (diffusion generation, 3D reconstruction, and interpolation) offloaded on the GPU-enabled cluster, Unity operates independently and remains responsive. In case of missing or late interpolated frames, Unity loops from the initial frame after the animation has ended.

Briefly, Unity is the real-time lightweight visualizer of the 3D facial animation pipeline. It takes the output of the offline computation stages and renders smooth expression transitions on a personalized 3D avatar. This decouples rendering from computation and enables both interactive performance and high visual quality without overburdening the front-end application.

Real-Time Considerations: One of the core implementation challenges was to make the entire pipeline run efficiently and in (near) real-time, despite spanning multiple processing-intensive steps and heterogeneous devices. I achieved this by parallelizing tasks across two GPUs on the Puhti supercomputer and a local Apple Mac M1 workstation, connected via network storage and automated file synchronization.

Figure 7 provides an overview of this automation pipeline. The system is divided into three components: 2D expression generation (on GPU 0), 3D reconstruction (on GPU 1), and animation and rendering (on Mac M1). Each module operates independently but synchronizes through shared directories, allowing asynchronous yet coordinated execution.

On Puhti, GPU 0 is responsible for handling input prompts and running the fine-tuned DiffEdit model for generating 2D expression images given a neutral input. Once the image has been generated, a skin tone matching step is employed to ensure visual consistency with the original image. The resulting final 2D output is stored in a shared directory accessible by other nodes.

At the same time, GPU 1 runs a custom DECA script that continuously monitors this shared directory. As soon as it identifies a new expression image, it performs

monocular 3D reconstruction and generates a textured 3D mesh (OBJ + UV texture) and stores it again to the same common space. Because the 3D reconstruction and image generation processes run in parallel and in separate GPU memory, this architecture allows the system to have a throughput of approximately one full frame per 2 seconds—halving the latency from a sequential setup.

The Mac M1 is the last node in the pipeline and handles the animation and visualization phases. The Mac M1 is connected to the Puhti cluster through SSH and has an `rsync` daemon running continuously to synchronize the shared output directory. When new 3D assets (texture and OBJ) are imported, the Mac performs two operations simultaneously: (1) running a high-frequency boosting script to improve the texture, and (2) performing interpolated meshes and textures for animation with a custom interpolation script. These in-between frames are then imported into Unity where they get rendered in real time as animation. While Unity is displaying the current frame, the pipeline can already be calculating the next expression, making the user interaction smooth.

All components of the system run independently, although they communicate with each other using solid, low-latency file watching and transferring methods. Scripts are structured so that prompts entered in the terminal on Puhti trigger the generation process, and each new file cascades through the system automatically, with minimal manual intervention. This design ensures robustness and scalability, as failures in one module do not halt the rest of the system.

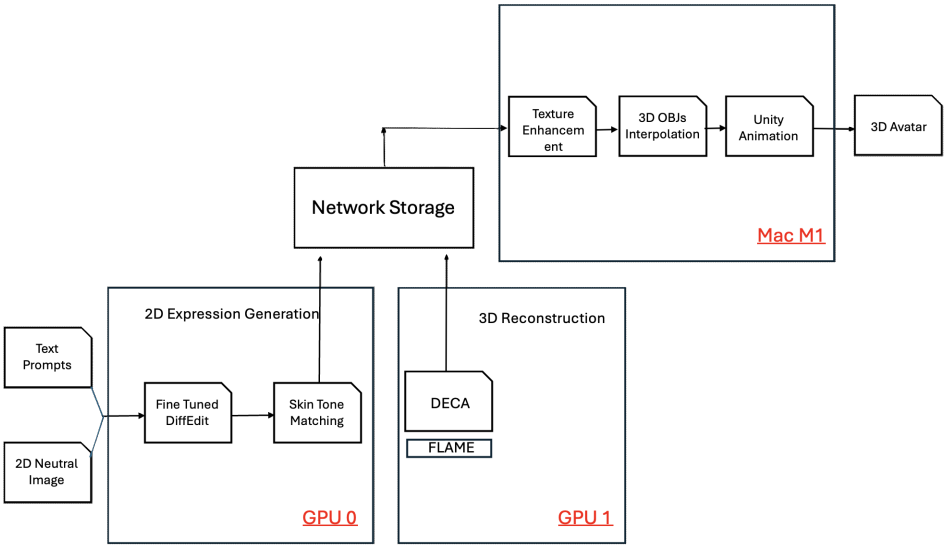

Figure 7. Automation Pipeline

To streamline the pipeline, I structured the execution into parallel components synchronized through shared directories and file polling. The sequence proceeds as follows:

- The user enters a new text prompt (e.g., `"happy face"`) in the terminal running on GPU 0. This input is queued by the DiffEdit script, which listens for user input in real-time.

- DiffEdit uses the provided prompt along with the fixed neutral image to generate a new 2D image reflecting the desired expression. Once generation is complete, the result is passed through a skin tone matching script for histogram-based color correction to maintain consistency with the original input image.
- The corrected 2D expression image is saved to a shared directory accessible by GPU 1. DECA, running on GPU 1 in parallel, continuously monitors this shared folder. Upon detecting a new image, it reconstructs the corresponding 3D face and generates the mesh (.OBJ) and texture files.
- The DECA outputs are written back to the shared folder. On the local Mac M1 machine, an `rsync` service synchronizes this directory over SSH, retrieving any new .OBJ and texture files as soon as they appear from the remote supercomputer.
- Upon detecting new avatars, the Mac triggers the texture enhancement module, improving visual fidelity by sharpening and enhancing the face texture.
- The enhanced outputs are then passed to the interpolation module, which calculates intermediate frames between the last rendered expression and the new one. These interpolated frames are saved as sequentially named files.
- Unity continuously monitors the local output directory. When it detects a new sequence of interpolated frames, it appends them to the current animation timeline, blending the new expression into the running animation seamlessly.
- Meanwhile, the DiffEdit script returns to a listening state, ready to accept the next prompt. DECA also remains idle but alert, watching for the next 2D image. The Mac-side scripts continue monitoring for new 3D files, completing the loop.

Through this parallel and asynchronous structure, the entire system maintains a seamless performance. Each component is optimized for its hardware: intensive neural network processing on the cluster GPUs, fast image processing on the CPU, and rendering on the Unity GPU. The user can continue animating indefinitely without ever waiting for the next step by hand; backend concurrency such that when the user is viewing one transition, the next transition might already be calculating in the background. Distributed configuration (remote HPC + local machine) did introduce synchronization complexity, but it also allowed me to leverage specialized hardware on specific tasks (e.g., GPUs for AI tasks, keeping the local GPU free for graphics). This is a common scenario for high-end usage where cloud or clustering resources are used to process AI and a client device is used for display.

In short, my strategy combined state-of-the-art neural techniques (DiffEdit, LoRA fine-tuning, DECA reconstruction) with careful systems engineering (parallel computation, file-based communication, and integration with Unity) to create a real-time 3D face expression animation pipeline.

4. EXPERIMENTS

In this section, I will compare the performance of each stage within the suggested pipeline. The comparison is organized around the pipeline's core modules, explaining how each module adds to real-time and realistic avatar expression animation. I will measure both the qualitative and quantitative aspects, such as expression realism, identity preservation, texture consistency, and system speed. By examining each module independently – from the 2D text-based inpainting to the final 3D rendering – I can demonstrate that each step does indeed work and is necessary for my thesis's ultimate goal. Below, I present results for each module and why these evaluations are crucial to the final animated avatar.

4.1. Text-To-Image Inpainting (DiffEdit + LoRA)

The first module produces a 2D face image with a new expression given a neutral face and a text prompt. I tested how well the DiffEdit [8] + LoRA [33] inpainting step can synthesize the target expression without losing the subject's identity. This is essential since any identity drift here would carry through the pipeline, compromising the realism of the output avatar.

For visual comparison, I conducted a qualitative test in which a neutral face image was first passed through the Stable Diffusion DiffEdit model without any fine-tuning. The same image was then run through the fine-tuned model. As is evident from Figure 8, the output generated by the fine-tuned model has substantially improved expression accuracy and consistency while retaining the identity of the subject even better.

Furthermore, Figure 9 shows other outputs generated by the fine-tuned DiffEdit model on other test subjects. Each row depicts a neutral input face followed by three edited outputs showing angry, happy, and sad expressions. The outputs qualitatively display the model's ability to produce semantically correct expression changes while maintaining the subject's identity, face contour, and background.

To assess the quality of the expression editing step quantitatively, I employed a collection of widely used image similarity measures. Every similarity measure scores a different aspect of performance:

- **Cosine Similarity** — measures to what extent the facial identity is preserved by quantifying deep face embeddings between real and generated images.
- **Structural Similarity Index (SSIM) [39]** — measures image quality based on structural characteristics such as luminance, contrast, and texture consistency.
- **Peak Signal-to-Noise Ratio (PSNR) [40]** — quantifies pixel-level image distortion, with higher numbers reflecting greater fidelity.

As can be seen from Table 1, cosine similarity between the original neutral face and manipulated faces was fairly high (particularly for the happy and sad ones), which indicates that the identity of the subject remained largely intact throughout editing.

Structural similarity (SSIM), presented in Table 2, was always above 0.86, which ensured that face geometry and texture were not deteriorated and no structural artifacts were introduced.

Table 3 shows that the Peak Signal-to-Noise Ratio (PSNR) scores ranged from 34.42 dB to 39.41 dB, showing minimal low-level distortion.

Collectively, these results verify that DiffEdit with fine-tuning using LoRA can generate perceptually strong expression changes with high visual quality and subject identity, as shown also in Figure 8 and Figure 9.

Additionally, the LPIPS (Learned Perceptual Image Patch Similarity) [38] score was also calculated to evaluate perceptual dissimilarity between the generated and neutral facial expression images. LPIPS compares high-level feature activations from a pretrained deep network to estimate visual perception difference between two generated images. In the case of this thesis, the LPIPS scores were relatively low (typically around 0.02 and 0.03), which aligns with the fact that facial expression edits tend to be localized and subtle. Low scores mean that while the model induces semantically meaningful changes (e.g., frown, smile), it does so with minimal perceptual disruption to the overall face shape — desirable in identity-preserving expression synthesis.

Moreover, it also gives a direction for future work, working towards better LPIPS score and better expression changes.

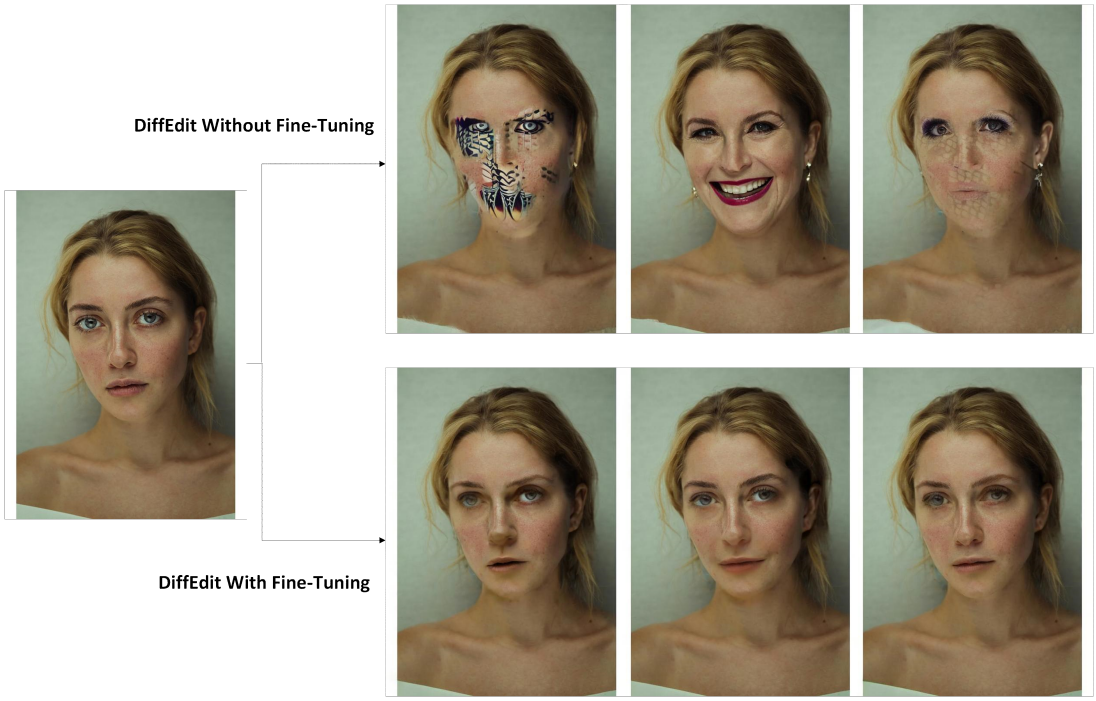

Figure 8. With & Without Fine-tuning Results

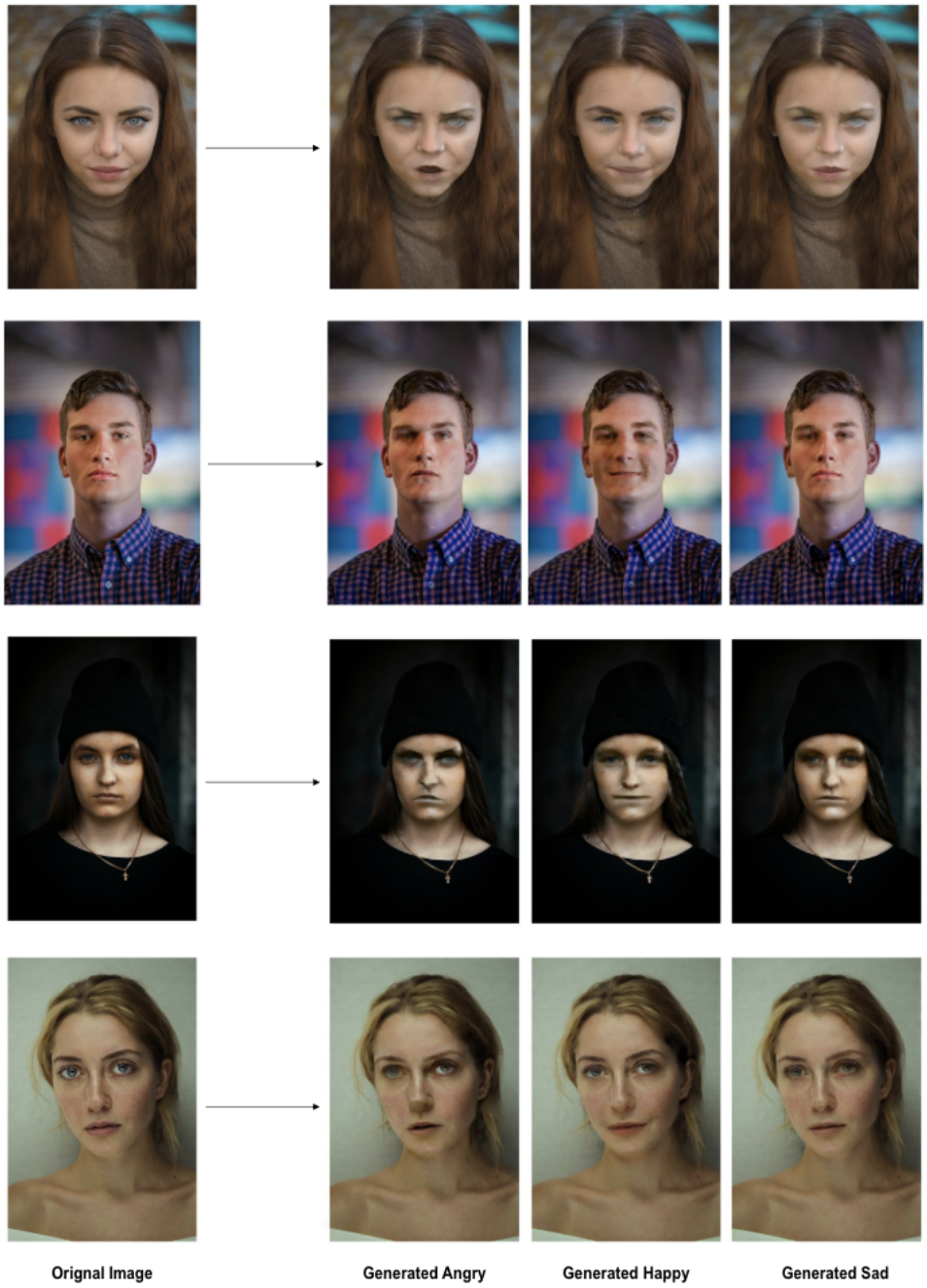

Figure 9. DiffEdit Outputs

Table 1. Cosine Similarity Between Generated Expression Images and Original Neutral Images for Different Subjects

Subject	**Angry Expression**	**Happy Expression**	**Sad Expression**
Image_1	0.5793	0.8211	0.8410
Image_2	0.7782	0.6812	0.9195
Image_3	0.6026	0.5796	0.7402
Image_4	0.7399	0.8318	0.8846

Table 2. SSIM Scores between Neutral and Generated Expression Images

Subject	Angry Expression	Happy Expression	Sad Expression
Image_1	0.8685	0.8724	0.8749
Image_2	0.9563	0.9499	0.9599
Image_3	0.9193	0.9215	0.9242
Image_4	0.9037	0.8957	0.9031

Table 3. PSNR (Peak Signal-to-Noise Ratio) between Neutral and Generated Expression Images (in dB)

Subject	Angry Expression	Happy Expression	Sad Expression
Image_1	35.98	36.05	36.53
Image_2	34.72	34.42	35.49
Image_3	39.37	38.28	39.41
Image_4	36.90	37.08	37.96

4.2. Skin Tone Matching

After expression editing, the second module solves skin tone inconsistency between the reconstructed face and the input neutral image. This is solved through a localized histogram matching process. Starting off, a center facial area is cropped to estimate the average face skin tone in LAB color space. Applying a perceptual distance thresholding afterwards generates a binary mask which identifies pixels in the constructed image resembling the tone of the central facial region. Histogram matching is done on these pixels, modifying the color distribution to correspond to that of the original reference image. Blending between the original pixel and corrected pixel values is facilitated by an intensity parameter to generate full or even partial correction.

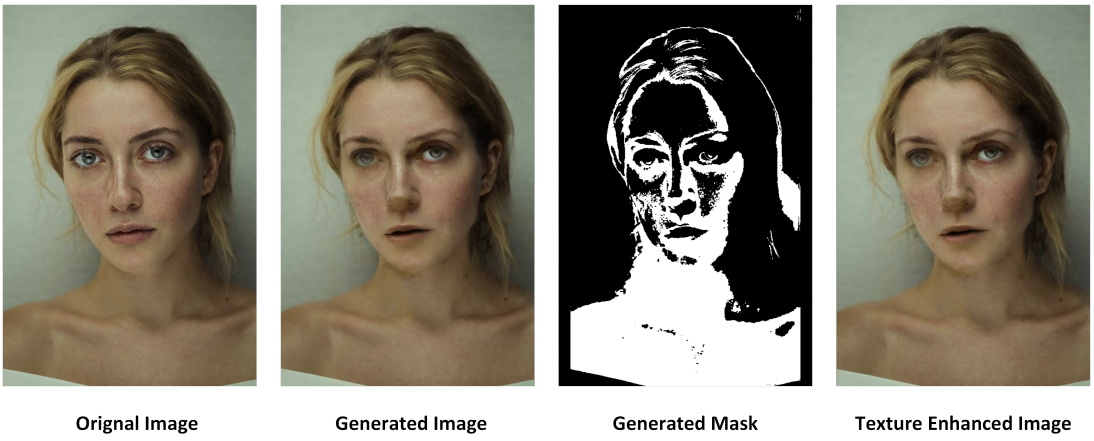

Figure 10. Skin Tone Matching

The process also generates visual diagnostics, including a binary mask of areas of pixels matched up and a heatmap of differences in pixels marking where the adjustments were made.

Figure 10 displays these outputs. Even though the corrected image at the end may appear visually very close to the uncorrected image for small differences in tone, the heatmap and mask confirm that the model is uniformly making selective adjustments. This correction module, which is light-weight, provides perceptual consistency between expression frames while preserving the identity of the subject. Table 4 further strengthens this histogram approach as I can see an increase in the similarity scores before and after applying the matching technique.

Table 4. Cosine Similarity Before and After Skin Tone Matching

Subject	**Before**			**After**		
	Angry	Happy	Sad	Angry	Happy	Sad
Image_1	0.5793	0.8211	0.8410	0.5858	0.8236	0.8375
Image_2	0.7782	0.6812	0.9195	0.7805	0.6902	0.9185
Image_3	0.6026	0.5796	0.7402	0.6185	0.5518	0.7363
Image_4	0.7399	0.8318	0.8846	0.7401	0.8427	0.8846

4.3. 3D Reconstruction With DECA

The DECA-based reconstruction [9] can effectively transform the edited 2D facial image into a high-fidelity 3D avatar with identity as well as expression retained. As shown in Figure 11, the reconstructed meshes possess correct geometry and identity features that are consistent across expressions. Each avatar not only records the expression described in the prompt (e.g., angry, happy), but also reproduces the individual's facial structure, such as jawline shape, nose structure, and location of eyes. This consistency is important for those uses where avatars should be identifiable when shown in varying emotional expressions.

Qualitatively, the models created by DECA show smooth geometry and visually consistent expressions, as indicated by the coherent head poses and natural-looking deformations. Even without ground-truth visibility for 3D scans, the rendered avatars closely resemble their 2D origins. The outcomes show that identity preservation during the DiffEdit phase also carries over to 3D, and DECA can maintain this level of fidelity throughout reconstruction. Furthermore, the reconstructed avatars can be animated and rendered for downstream usage, which is an ideal decision for real-time use.

Figure 11. DECA Results

4.3.1. 3D Interpolation

With personalized 3D avatars reconstructed, I then tested the interpolation module's performance at smoothly animating the expression. The module interpolates linearly between two expression meshes (such as neutral to anger) through interpolating vertex positions and 3D face parameters. The goal is to achieve visually continuous artifact-free motion, identity, and realism preservation during transition.

The interpolation method produces smooth and realistic-looking facial expressions. As Figure 12 illustrates, every in-between frame is a plausible expression state on the same face, gradually transitioning from origin to the target emotion. The avatar's geometry warps uniformly—without any jumps or artifacts, preserving subtle facial details and maintaining temporal consistency. The transitions are expressive but under-control, making the emotional transitions look natural.

This ongoing evolution of expressions is a critical component for the realism of the final animation. Interpolation operates in low-dimensional parameter spaces and does not involve any significant computational overhead, enabling real-time playback at 60 FPS. Overall, the module is capable of smoothly connecting static expression states with dynamic animation, rendering the avatar responsive and smooth in real-time scenarios.

Figure 12. Interpolation Results

4.4. Texture Enhancement

Once the 2D face image is processed and the texture mapped onto the 3D mesh by DECA, I then add an efficient boosting step to enhance facial details before visualization. Rather than employing complex super-resolution models, I picked a simple yet effective high-frequency boosting technique based on Gaussian blurring.

This method adds detail of texture by emphasizing high-frequency information (e.g., wrinkles, microstructure of the skin) that contributes to realism but can be removed in 3D reconstruction or diffusion generation. Specifically, I subtract a blurred version from the original to have high-frequency information remaining, resize it, and place it back into the texture. This leaves me with a texture map that has improved sharpness but preserves lighting and color of the original image but adds perceptual detail.

This enhancement causes near to zero computational overhead which makes it much suitable for real-time pipelines.

Figure 13 shows the improved visual detail of DECA-rendered avatars after applying this texture sharpening method. Intricate details such as the crease around the nose, lip contours, and brow texture become more prominent, without influencing global overall appearance.

Figure 14 also demonstrates the benefit in the animated transition case. As opposed to unsharpened textures being subtly smeared across frames, the sharpened sequence preserves detail continuity and thereby adds to overall realism during facial motion.

This lightweight method provides a fast and visually convincing enhancement to otherwise flat 3D textures, playing a substantial role in final avatar quality without trading off real-time performance.

The final pipeline implementation and unity animation can be seen here [41].

Figure 13. Texture Enhanced DECA Results

Figure 14. Enhanced Interpolation Results

5. DISCUSSION

The proposed text-to-3D avatar animation system demonstrates a feasible pipeline for generating expressive 3D digital faces from neutral images. By leveraging a fine-tuned diffusion model for 2D inpainting and DECA [9] for 3D reconstruction, the approach can translate simple text prompts (e.g. “angry face”) into visible changes in a 3D avatar’s expression. However, the implementation revealed several challenges and limitations that must be addressed to improve the system’s practicality and fidelity. Foremost, there is an inherent trade-off between the strength of the induced facial expression and the preservation of the person’s identity. Pushing the generative model to produce more pronounced expressions sometimes altered facial geometry or texture enough to diminish identity fidelity, whereas constraining the edits to protect identity often led to more subtle, less expressive results. This tension between expression intensity and identity consistency is well documented in related work, and it was evident in my results as I fine-tuned the model (via DiffEdit [8] and LoRA [33]) to maintain the subject’s key features. It is still difficult to have high-fidelity expressions without "identity leakage" because very strong expression cues actually end up altering typical facial contours, while maintaining identity too strictly kills expressions. Getting the correct tradeoff among these components involved meticulous prompt development and model parameter tuning, and even with that there was sometimes a bit of compromise either in the realism of expression or the accuracy of identities.

I also observed that maintaining texture and lighting consistency across frames is a fragile aspect of the pipeline. Because the 2D diffusion model edits each expression on the neutral image independently, slight inconsistencies in the output can appear when generating a sequence of expressions for animation. For example, an edited picture can introduce a small change to skin tone, specular highlight, or hair texture that will not be replicated by another frame, even though identity and surroundings are intended to be identical. When these frames are reconstructed to 3D and displayed sequentially, the result may be slight flicker or an impression that the face of the avatar is "shifting" unexpectedly. This issue of temporal or cross-frame consistency is a known shortcoming of image-based generation methods applied to animation. Even though I used interpolation in Unity to smooth the transitions, the underlying frame-by-frame texture differences could still lead to visible artifacts (e.g. a mole appearing and disappearing, or a change in eyebrow texture) that undermined realism. The problem is essentially that my method lacks an explicit temporal constraint or feedback mechanism to enforce coherence across frames. Without a mechanism akin to a temporal diffusion model or a consistent texture map, each edit is done in isolation. As a result, the avatar’s appearance can drift over time if one isn’t careful. Moreover, the DECA-based reconstruction itself, while robust for capturing overall facial shape and expression, relies on a morphable model that may not capture all the high-frequency details present in the edited 2D image. This can slightly blur or homogenize textures when moving to 3D, and if the 2D edits between frames vary in those high-frequency details, the 3D avatar might emphasize the discrepancies. In summary, ensuring that the avatar’s texture and expression remain stable and continuous throughout an animation is an open challenge. Approaches from recent research on diffusion-based video editing explicitly address temporal consistency to mitigate flicker, and such ideas could inform improvements to my pipeline (for instance, by adding a temporal

consistency loss or using a sequential model that propagates texture information frame to frame).

Despite these limitations, the core concept of text-driven avatar animation opens some interesting avenues for future work. One of these is the incorporation of multi-modal input for driving expressions. A user would need to type in a desired expression (e.g., "a surprised face with a bit of a smile"), but human communication also occurs through voice, intonation, and even physiology signals. Adding audio or speech analysis would enable the system to create facial animations in synchronization with the mood and tone of a speaker in real-time. A virtual therapy avatar, for instance, would automatically portray empathetic responses or reflect a patient's emotional state when receiving voice input, making the interaction more interactive and compassionate. Similarly, in a digital assistant application, the face of the avatar can respond to both what is said (the text itself) and the manner in which it is said (emotion in tone) to generate more human-like and contextually relevant responses. A multi-modal expansion would involve combining the text-to-image model with an emotion detection model from speech or a viseme generation model (for lip-syncing), with facial movement coordinated with speech. It presents its own set of challenges – primarily, coordinating timing and level of effort across modalities – but might well significantly expand the system's usefulness to areas requiring interactive, affectively responsive digital humans.

A further promising extension is to transition from discrete expression prompts to a more continuous and manipulable expression space. In the present architecture, the user normally prescribes categorical expressions ("happy", "sad", "surprised", etc.), and the model synthesizes separate frames for each. In future research, one may offer a means to blend or interpolate between expressions, enabling more precise control of the intensity and subtlety of the avatar's emotions. This can be done by introducing continuous control parameters (for example, sliders for arousal and valence, or explicit blendshape coefficients) in addition to the text prompt, or by training the diffusion model on a conditioned range of expressions (for example, 0% to 100% happy). The advantage of ongoing control would be more graceful transitions and the potential to express subtle emotional blends – a character could be 70% happy and 30% surprised and have a surprised delight look that a word prompt would not provide. This would be an enhancement over animated storytelling and game situations, where subtle emotional expression is important. It would allow writers to author individual emotional beats, and game characters to react in graduated manner to what the player is doing. Implementing this would likely involve extending the model's training data to intermediate expressions or utilizing interpolation within the expression latent space. Keeping that identity intact throughout such mixed states will remain important, but with careful tuning, the avatar could transition expression smoothly without losing its identity.

Outside of the face, an obvious but challenging next step is to apply the system to full-body avatars. Realistic digital humans can communicate not just with facial expressions, but with body posture and gesture. A next-generation system might be able to take a text prompt such as "the character is excited and waves their hand" and generate a full-body 3D animation that matches both facial expression and physical gesture. This would involve integrating my present method into text-to-motion or pose generation algorithms. One potential method is to employ a parametric human

model (like those utilized in motion capture animation) and animate it with diffusion-generated pose predictions or inverse kinematics from the text prompt. There are numerous challenges here: getting the face and body gestures coordinated, maintaining the identity in body shape and gait as I currently do with the face, and avoiding the uncanny valley effect as complexity is added. But success here would greatly expand the range of uses. For example, full-body expressive avatars would be used in social virtual reality or in sophisticated video games, where every mannerism of a character (posture, hand gestures, and facial expressions) reacts to story events or user input. Even telecommunication or telepresence would use a complete avatar that is able to smile, shrug, or gesture to enable interacting digitally to become more natural and interactive. While my current work lays the foundation with facial animation, progressing to a full-avatar animation system is the logical next step to digitization as realistic as possible.

Lastly, pushing the realism of the 3D expressions and avatar fidelity overall continues to be a desirable objective. DECA-based reconstruction does allow for an easy method to obtain a 3D mesh with expressions from a single image but does place a limit on fidelity – its underlying face model might not record each wrinkle or precisely the nuances in how muscle and skin move. Future work can employ higher-fidelity facial anatomy models or leverage neural rendering methods for increased realism. One can, for instance, employ higher resolution textures (possibly by combining the diffusion model output with a super-resolution network trained on facial details) so that finer details like eyelashes, pores, or dimples are transferred to the 3D avatar. Another idea is to retouch the expression transfer by adjusting the blendshape parameters of the 3D model to better match the edited picture, possibly through an optimization process that minimizes the difference between the 3D face rendering and the 2D diffusion output. This would correct any small mistakes introduced by the reconstruction process. Also, for dynamic realism, techniques that take into account transient expression details – i.e., exposure of veins while laughing or slight reddening of the skin – can be taken into consideration, although these venture into the realm of on-the-fly texture generation and may take more data (or even a switch to a video-based solution instead of single-image). Increasing 3D expression realism is not just an artistic goal; it has a very direct effect on how believable and relatable the avatar becomes. In uses like simulation training with digital humans or virtual therapy, where empathy and trust are key, having an avatar that emotively expresses and physically moves in a human-like way is crucial. My present findings are a move in that direction, with scope for finer replication of real facial behavior under emotion in the avatar's expressions.

In conclusion, this thesis demonstrates the possibility of integrating state-of-the-art image diffusion models and 3D face reconstruction for generating text-based animatable avatars and the practical challenges of doing so. By overcoming the limitations described herein, more evenly balancing expression and identity, minimizing system latency, enforcing consistency, and increasing the range and realism of the avatar, next-generation systems can be substantially more powerful and flexible. These advances will spawn more general uses in everything from interactive entertainment to corporate communication. Content creators, for instance, could use a more advanced system to power animated storytelling, quickly prototyping characters that respond decisively to actions in a script. In therapy or education, a

therapist or teacher can use a digital avatar that empathetically responds to patients or students, making interactions more engaging and personalized. In social virtual reality and digital human applications, users can be embodied in avatars that convey their expressions exactly as they talk or collaborate, enhancing the sense of presence. The concept of text, vision, and 3D animation technology intersecting therefore promises to enhance the way virtual characters are created and interacted with. Further research into making systems like mine more faithful and responsive will lead us closer to realistic, real-time animated avatars that can co-exist with human players in games, narratives, and environments.

6. CONCLUSION

This thesis demonstrated a real-time pipeline for the animation of high-fidelity 3D avatars from a single neutral image and a text prompt. With the synergy of generative diffusion models and 3D face reconstruction, the system permits natural language control of high-fidelity expression in a seamless way. A Stable Diffusion model fine-tuned (with DiffEdit and LoRA) adjusts the facial expression while preserving identity, and DECA reconstructs the transformed image into a textured 3D mesh. Additions such as histogram-based skin color correction and texture sharpening add realism, and mesh and texture interpolation enables smooth expression change from one expression to another. The rendering is done in Unity, enabling real-time visual feedback and user interaction.

The pipeline's modular and GPU-parallel design reaches responsiveness and scalability, and it is well suited for applications in virtual avatars, real-time communications, and interactive media. Each stage, from expression generation to rendering into 3D, was optimized both for speed and for quality, demonstrating the promise of a user-directed, text-to-3D facial animation system.

This work establishes an empirical foundation for future systems that may incorporate more extensive modalities like speech-controlled animation, head pose control, or whole-body incorporation. In its bridge between 2D generative models and controllable 3D products, the approach makes it possible to allow accessible, expressive digital humans in real-time environments.

7. REFERENCES

[1] Egger B., Smith W.A.P., Tewari A., Wuhrer S., Zollhoefer M., Beeler T., Bernard F., Bolkart T., Kortylewski A., Romdhani S., Theobalt C., Blanz V. & Vetter T. (2020) 3d morphable face models—past, present, and future. ACM Transactions on Graphics (TOG) 39, pp. 1–38. URL: https://doi.org/10.1145/3386569.3392487.

[2] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. & Bengio Y. (2014) Generative adversarial networks. arXiv preprint arXiv:1406.2661 URL: https://arxiv.org/abs/1406.2661.

[3] Yang L., Zhang Z., Song Y., Hong S., Xu R., Zhao Y., Zhang W., Cui B. & Yang M.H. (2022) Diffusion models: A comprehensive survey of methods and applications. arXiv preprint arXiv:2209.00796 URL: https://arxiv.org/abs/2209.00796.

[4] Rombach R., Blattmann A., Lorenz D., Esser P. & Ommer B. (2022) High-resolution image synthesis with latent diffusion models. arXiv preprint arXiv:2112.10752 URL: https://arxiv.org/abs/2112.10752.

[5] Ramesh A., Pavlov M., Goh G., Gray S., Voss C., Radford A., Chen M. & Sutskever I. (2021) Zero-shot text-to-image generation. arXiv preprint arXiv:2102.12092 URL: https://arxiv.org/abs/2102.12092.

[6] Poole B., Jain A., Barron J.T. & Mildenhall B. (2022) Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 URL: https://arxiv.org/abs/2209.14988.

[7] Mildenhall B., Srinivasan P.P., Tancik M., Barron J.T., Ramamoorthi R. & Ng R. (2020) Nerf: Representing scenes as neural radiance fields for view synthesis. arXiv preprint arXiv:2003.08934 URL: https://arxiv.org/abs/2003.08934.

[8] Couairon G., Verbeek J., Schwenk H. & Cord M. (2022) Diffedit: Diffusion-based semantic image editing with mask guidance. arXiv preprint arXiv:2210.11427 URL: https://arxiv.org/abs/2210.11427.

[9] Feng Y., Feng H., Black M.J. & Bolkart T. (2021) Learning an animatable detailed 3d face model from in-the-wild images. ACM Transactions on Graphics (TOG) 40, pp. 88:1–88:13. URL: https://doi.org/10.1145/3450626.3459936.

[10] Radford A., Kim J.W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., Clark J., Krueger G. & Sutskever I. (2021) Learning transferable visual models from natural language supervision. International Conference on Machine Learning (ICML) URL: https://arxiv.org/abs/2103.00020.

[11] Patashnik O., Wu Z., Shechtman E., Cohen-Or D. & Lischinski D. (2021) Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2085–2094. URL: https://arxiv.org/abs/2109.08379.

[12] Xia W., Yang Y., Xue J.H. & Wu B. (2021) Tedigan: Text-guided diverse face image generation and manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2256–2265. URL: https://arxiv.org/abs/2012.03308.

[13] Fu W.W., Gong W.J., Yu C.Y., Wang W. & Gonzlez J. (2025) Facial expression generation from text with faceclip. Journal of Computer Science and Technology 40, pp. 359–377. URL: https://doi.org/10.1007/s11390-024-3661-z.

[14] Brooks T., Holynski A. & Efros A.A. (2023) Instructpix2pix: Learning to follow image editing instructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18392–18402. URL: https://arxiv.org/abs/2211.09800.

[15] Aneja S., Thies J., Dai A. & Niessner M. (2023) ClipFace: Text-guided editing of textured 3d morphable models. In: ACM SIGGRAPH 2023 Conference Proceedings, pp. 1–11. URL: https://arxiv.org/abs/2212.01406.

[16] Hwang S., Hyung J., Kim D., Kim M.J. & Choo J. (2023) Faceclipnerf: Text-driven 3d face manipulation using deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3469–3479. URL: https://arxiv.org/abs/2307.11418.

[17] Li T.B., Béatrice & Black M.J. (2017) Flame: Learning a model of facial shape and expression from 4d scans. ACM Transactions on Graphics (TOG) 36, pp. 194:1–194:17. URL: https://dl.acm.org/doi/10.1145/3130800.3130813.

[18] Daněček R., Black M.J. & Bolkart T. (2022) EMOCA: Emotion driven monocular face capture and animation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 20279–20290. URL: https://arxiv.org/abs/2204.11312.

[19] Bai Z., Cui Z., Liu X. & Tan P. (2021) Riggable 3d face reconstruction via in-network optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6216–6225. URL: https://arxiv.org/abs/2104.03493.

[20] Zhuang Y., Zhu H., Sun X. & Cao X. (2022) MoFaNeRF: Morphable facial neural radiance field. In: Computer Vision – ECCV 2022, Lecture Notes in Computer Science, vol. 13663, Lecture Notes in Computer Science, vol. 13663, pp. 268–285. URL: https://arxiv.org/abs/2112.02308.

[21] Chen X., Mihajlovic M., Wang S., Prokudin S. & Tang S. (2024) Morphable diffusion: 3d-consistent diffusion for single-image avatar creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10359–10370. URL: https://arxiv.org/abs/2401.04728.

[22] Ha S., Kersner M., Kim B., Seo S. & Kim D. (2020) MarioNETte: Few-shot face reenactment preserving identity of unseen targets. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, vol. 34, pp. 10893–10900. URL: https://arxiv.org/abs/1911.08139.

[23] Ren Y., Li G., Chen Y., Li T.H. & Liu S. (2021) Pirenderer: Controllable portrait image generation via semantic neural rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 13759–13768.

[24] Xu C., Zhang J., Han Y., Tian G., Zeng X., Tai Y., Wang Y., Wang C. & Liu Y. (2022) Designing one unified framework for high-fidelity face reenactment and swapping. In: Computer Vision – ECCV 2022, Lecture Notes in Computer Science, vol. 13675, Lecture Notes in Computer Science, vol. 13675, pp. 54–71. URL: https://dl.acm.org/doi/10.1007/978-3-031-19784-0_4.

[25] Zhang J., Li X., Wan Z., Wang C. & Liao J. (2022) FDNeRF: Few-shot dynamic neural radiance fields for face reconstruction and expression editing. In: SIGGRAPH Asia 2022 Conference Papers, pp. Article 12, 1–9. URL: https://arxiv.org/abs/2208.05751.

[26] Oorloff T. & Yacoob Y. (2023) Robust one-shot face video re-enactment using hybrid latent spaces of stylegan2. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV Workshops), pp. 20890–20900. URL: https://arxiv.org/abs/2302.07848.

[27] Deng Y., Yang J., Chen D., Wen F. & Tong X. (2020) Disentangled and controllable face image generation via 3d imitative-contrastive learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5154–5163. URL: https://arxiv.org/abs/2004.11660.

[28] Li L., Bao J., Yang H., Chen D. & Wen F. (2020) Advancing high fidelity identity swapping for forgery detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5074–5083. URL: https://ieeexplore.ieee.org/document/9156865.

[29] Deng J., Guo J., Niannan X. & Zafeiriou S. (2019) Arcface: Additive angular margin loss for deep face recognition. In: CVPR, pp. 4690–4699. URL: https://arxiv.org/abs/1801.07698.

[30] Ding Z., Zhang X., Xia Z., Jebe L., Tu Z. & Zhang X. (2023) Diffusionrig: Learning personalized priors for facial appearance editing. In: Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12736–12746. URL: https://arxiv.org/abs/2304.06711.

[31] Khan M.W., Jia M., Zhang X., Yu E., Shan C. & Musial-Gabrys K. (2025) InstaFace: Identity-preserving facial editing with single image inference. arXiv preprint arXiv:2502.20577 URL: https://arxiv.org/abs/2502.20577.

[32] Anand T., Garg A. & Mitra K. (2025) IP-FaceDiff: Identity-preserving facial video editing with diffusion. arXiv preprint arXiv:2501.07530 URL: https://arxiv.org/abs/2501.07530.

[33] Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L. & Chen W. (2021), Lora: Low-rank adaptation of large language models. URL: https://arxiv.org/abs/2106.09685.

[34] van der Walt S., Schönberger J.L., Nunez-Iglesias J., Boulogne F., Warner J.D., Yager N., Gouillart E., Yu T. & the scikit-image contributors (2014) scikit-image: image processing in python. PeerJ 2, p. e453. URL: https://peerj.com/articles/453.

[35] Rombach R., Blattmann A., Lorenz D., Esser P. & Ommer B. (2022), Stable diffusion v2.1. URL: https://huggingface.co/stabilityai/stable-diffusion-2-1, version 2.1 released by Stability AI and CompVis.

[36] Lundqvist D., Flykt A. & Öhman A. (1998) The karolinska directed emotional faces (kdef). Tech. Rep. CD ROM from Department of Clinical Neuroscience, Karolinska Institutet, Department of Clinical Neuroscience, Psychology section, Karolinska Institutet. URL: https://www.kdef.se/, available at: https://www.kdef.se/.

[37] CSC – IT Center for Science (2023), Puhti supercomputer. https://docs.csc.fi/computing/systems/puhti/. URL: https://docs.csc.fi/computing/systems-puhti/, accessed: 2025-05-24.

[38] Zhang R., Isola P., Efros A.A., Shechtman E. & Wang O. (2018) The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 586–595. URL: https://github.com/richzhang/PerceptualSimilarity.

[39] Wang Z., Bovik A.C., Sheikh H.R. & Simoncelli E.P. (2004) Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing 13, pp. 600–612. URL: https://doi.org/10.1109/TIP.2003.819861.

[40] Hore A. & Ziou D. (2010) Image quality metrics: Psnr vs. ssim. In: 2010 20th International Conference on Pattern Recognition, IEEE, pp. 2366–2369. URL: https://doi.org/10.1109/ICPR.2010.579.

[41] Syed A. (2025), From words to 3d faces: A real-time pipeline for avatar expression animation. URL: https://www.youtube.com/watch?v=275gBGp2yXM&ab_channel=AstronomerAbdurrehman, youTube video.

8. APPENDICES

APPENDIX 1 ALTERNATIVE APPROACHES

This appendix covers some of the alternative techniques that were being looked into during the research but never pursued. These efforts included a 3D spatial mapping concept, simpler 2D image models, diffusion-based generative methods, language model feature embeddings, and the FLAME face model. All of these were explored and tested but the availability of data, computational expense, or integration challenges prevented these methods from being implemented in the final system.

One strategy involved constructing a 3D mapping pipeline as shown in Figure 15, the goal was to capture spatial structure from input image and context from the text prompts. However, reliable three-dimensional reconstruction proved complex due to the lack of solutions available for text + image to 3D models. Idea for designing a deep learning model was also explored such that a model after taking the necessary inputs generate embeddings in such a way that can be directly passed to DECA, however, given the complexity of the problem and lack of training datasets, this approach was deemed impractical. This lead to the idea of incorporating purely 2D models as a simple workaround. These models required less computation and integrated easily with the existing pipeline, but their neglect of depth information resulted in suboptimal performance on tasks requiring spatial understanding.

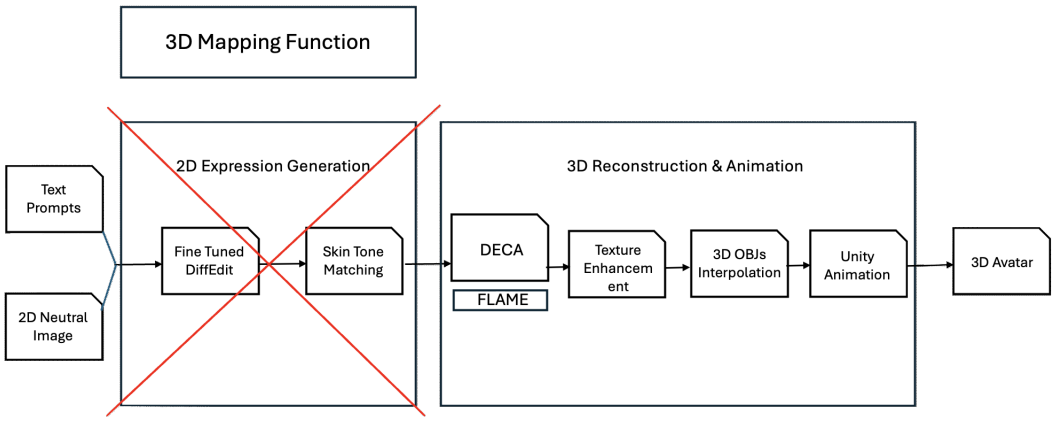

Figure 15. Alternate 3D Mapping Approach

Another idea was to incorporate semantic context using large language model (LLM) embeddings. I attempted to leverage pre-trained LLMs to generate contextual feature vectors for my data. While this approach provided richer semantic features, aligning the embeddings with DECA specific case for expression generation task proved challenging. Afterwards, this idea was dropped as well.

I then experimented with diffusion-based generative methods to improve output quality, models that could take the necessary inputs and provide an output that preserves the identity while promising realistic expressions. I tried several diffusion based models, Pix2Pix, stable diffusion (with manual masking), diffEdit based stable diffusion model. It was observed that out of the three only diffEdit provided promising results, however, it would need to be fine-tuned for it to work according to the desired objectives of this thesis.

Figure 16 shows the block which was replaced with these other models.

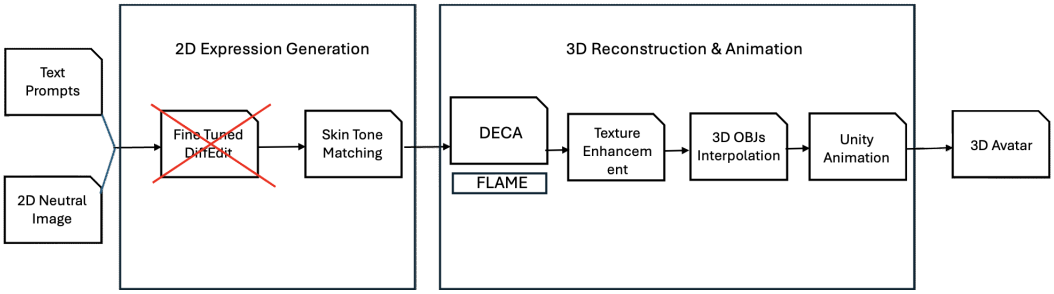

Figure 16. Alternate 2D Approaches

Additionally, Figure 17 shows the original image that was given to Instruct Pix2Pix model and the generated outputs.

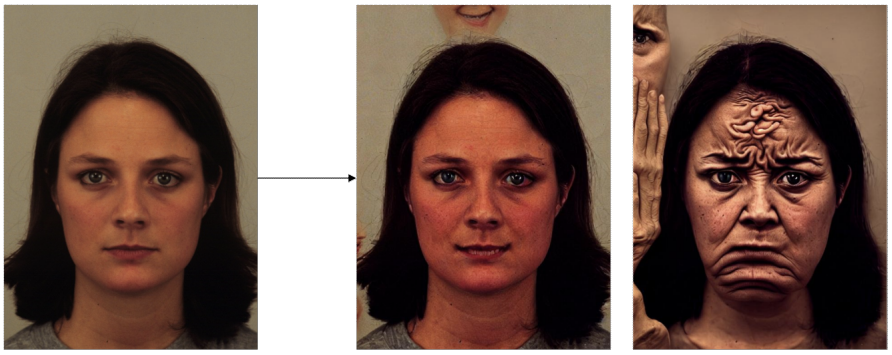

Figure 17. Instruct Pix2Pix Model Results

Initially, the 3D model used was FLAME and not DECA. Due to its detailed geometry and consistent 3D face reconstruction capabilities, it was chosen. However, manual tuning was required for the shape, pose, and expression parameters and for the reconstruction. This complexity of integrating FLAME into my workflow outweighed its potential benefits, and this approach was therefore excluded from the final design. Figures 18 and 19 show various outputs I got from FLAME.

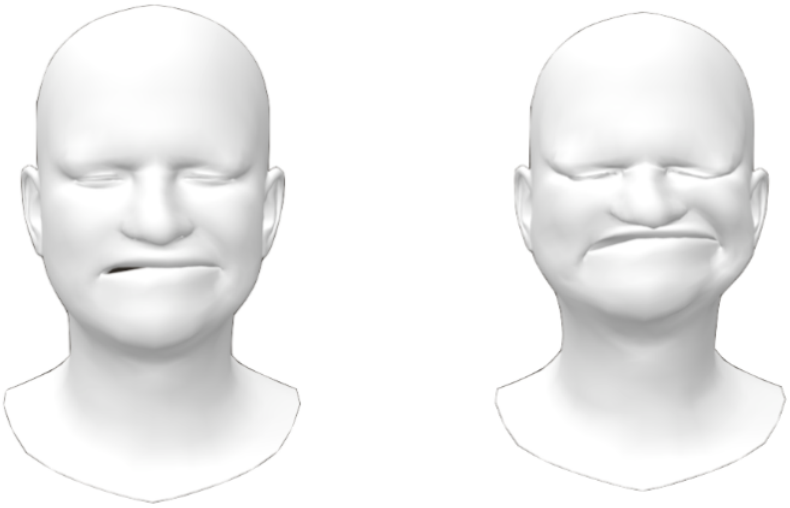

Figure 18. Flame Outputs (1)

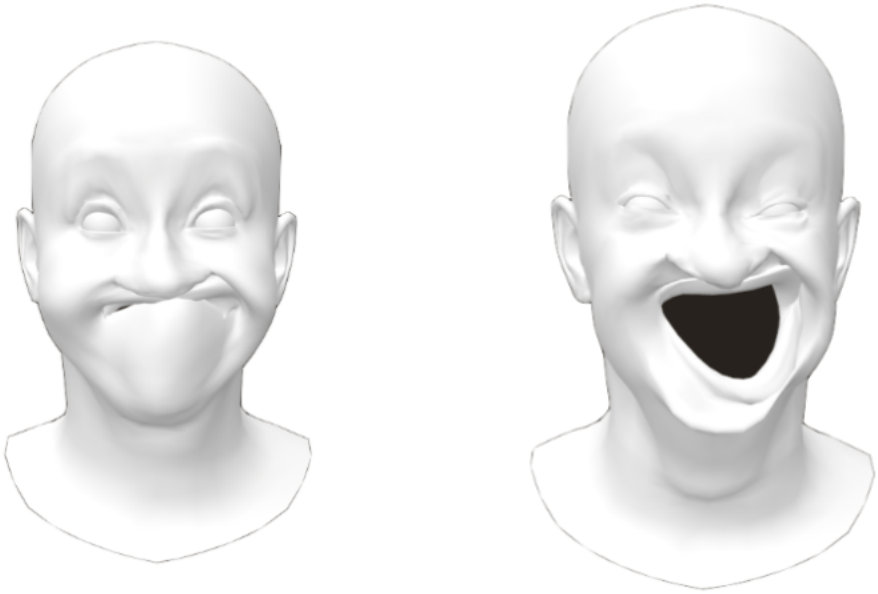

Figure 19. Flame Outputs (2)

APPENDIX 2 IMAGE ATTRIBUTIONS

This appendix lists the image sources used in this thesis. Images were obtained from two publicly available, non-commercial sources:

1. **Bodies in Motion (Scott Eaton)** – Reference images used for facial expression and pose analysis, under the Creative Commons BY-NC 4.0 license: https://www.bodiesinmotion.photo/expressions

2. **Unsplash** – Four images were used to create a composite reference figure:

   - Photo by Vlad Rudkov: https://unsplash.com/photos/a-woman-with-long-brown-hair-and-blue-eyes-Q_ZIfj3Ahro
   - Photo by Anastasia Vityukova: https://unsplash.com/photos/woman-in-black-shirt-wearing-black-beanie-and-gold-colored-necklace-with-pendant-kVpxpYh-zvw
   - Photo by David French: https://unsplash.com/photos/man-in-blue-and-white-checkered-dress-shirt-N0e6W2WDa5U
   - Photo by M_pxio: https://unsplash.com/photos/a-woman-with-blue-eyes-is-looking-at-the-camera-tGZAcWdAHLg