DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Text Guided Generation of Head Avatars

Christoph Daniel Höll

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Text Guided Generation of Head Avatars

Textgesteuerte Erzeugung von Kopf-Avataren

Author:	Christoph Daniel Höll
Supervisor:	Prof. Dr. Matthias Nießner
Advisor:	Shivangi Aneja
Submission Date:	15.07.2024

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

Munich, 15.07.2024 Christoph Daniel Höll

Acknowledgments

I would like to thank my advisor Shivangi Aneja, for the support, ideas, and critiques offered during this thesis. Especially maintaining our weekly meetings, even during stressful times, deserves my deep gratitude. This work is a result of many discussions and ideas. I would like to thank my supervisor, Prof. Dr. Matthias Nießner, for providing me with this opportunity to work on my Master's thesis at the Visual Computing Lab. I would also like to extend my thanks to Christoph Weiler and the whole technician team of the chair of the Visual Computing Lab for the provided hardware and the persistent support. Finally, I wish to thank my family and friends, especially my parents, for their support and encouragement.

Abstract

With the recent and rapid advancements in computer graphics and deep learning, it is now possible to create high-quality videos and even animation sequences from a single textual description. Currently, most of the research on generating motion sequences from text conditioning is directed to the full-body domain. In the facial domain, the number of works is very limited, and most aim to generate lip-synchronous facial movement with a provided audio input. This limitation is caused by a lack of a large and high-quality dataset matching textual descriptions and the corresponding motion sequences. With this work, we aim to take the first step towards generating facial motions solely from textual descriptions.

We begin by exploring 3D facial reconstruction from monocular images and videos. Here, we define several properties a monocular image must uphold to be reconstructed correctly in the 3D space and, based on these properties, propose a pipeline with which we can generate a large-scale 3D facial motion dataset.

We propose the first vector-quantized transformer-based sequence model to perform text-guided synthesis of human head avatars. In contrast to existing work on bodies, our method operates on a free mesh surface in the form of vertex displacements. By using the pre-trained Language-to-Image model CLIP, we leverage the large amount of (text, image) training data to improve our capabilities to handle unseen text prompts. Utilizing a Vector Quantized Variational Autoencoder enables our model to create diverse output while maintaining sequential cohesion through the transformer-based sequence model.

As no previous work aimed to generate facial motion from a textual description exists, we compare our model to models of the full-body motion domain adapted to and trained on our newly generated dataset. Here, we adopt the established metrics for the text-guided motion generation used often in this domain [1] [2] [3] [4]. We show that our method outperforms existing methods both qualitatively and quantitatively in the results chapter. Finally, we will provide some insights into the limitations and possible improvements of our work

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**

2 Thesis Overview **3**

3 Background **4**
3.1 3D Morphable Face Models 4
3.1.1 Basel Face Model 4
3.1.2 FLAME 6
3.2 Image-Text Models 8
3.3 Generative Models 9
3.3.1 Variational Autoencoders 10
3.3.2 Vector Quantized Variational Autoencoders 14
3.4 Transformers 15
3.4.1 General Structure 15
3.4.2 Positional Encoding 17
3.4.3 Attention Mechanism 20

4 Related Work **21**
4.1 Dataset 21
4.1.1 CelebV-Text 21
4.1.2 FaMoS 25
4.2 3D Tracking 27
4.2.1 DECA 27
4.2.2 MICA & Metrical Tracker 30
4.3 Text Guided Motion Generation 31
4.3.1 Human Motion Diffusion Model 31
4.3.2 ACTOR 34
4.4 Guided Facial Motion Generation 36

5 Dataset **39**
5.1 Data Selection Process 39
5.2 Processing Steps 43
5.2.1 Pre-Processing 43

5.2.2 Video Download . . . 43
5.2.3 3D Tracking . . . 45
5.2.4 Post-Processing . . . 45
5.3 Further Modifications . . . 46
5.4 Efficiency Improving Measures . . . 46
5.5 Visualizations . . . 46

6 Proposed Method **50**
6.1 Input Representation . . . 50
6.2 Model Overview . . . 51
6.2.1 Encoder . . . 51
6.2.2 Decoder . . . 53
6.3 Losses . . . 53

7 Results **55**
7.1 Implementation Details . . . 55
7.2 Baseline Experiments . . . 55
7.2.1 Qualitative Analysis . . . 56
7.2.2 Quantitative Analysis . . . 60
7.3 Ablation Studies . . . 61

8 Limitations & Future Work **63**

9 Final Remarks **65**

List of Figures **66**

List of Tables **71**

Bibliography **72**

1 Introduction

Facial animation has been challenging since the beginning of digital video generation and modification, from the early hand-drawn origins to today's sophisticated computer-generated imagery (CGI). One of animation's most fascinating and complex tasks is the realistic portrayal of human motion. Especially in the facial region, the so-called Uncanny Valley [5] effect poses a serious challenge. It describes the phenomenon "that a person's response to a humanlike robot would abruptly shift from empathy to revulsion as it approached, but failed to attain, a lifelike appearance" [5]. However, it is observed not only in robotics but also in CGI and animations, posing a tough challenge. Nowadays, no motions are fully animated if they are aimed to be as realistic as possible; instead, they are captured from an actual human through motion-capturing technologies. But this requirement increases the cost financially and in terms of time manifold. The financial requirement can, especially in lower-budget productions like independent film artists or indie game studios, result in the unfeasibility of the project as a whole. However, realistic and cheap animation is not only desirable from an artistic or entertainment point of view but can also provide benefits in day-to-day life. In virtual reality, realistic head animations can enhance the immersive experience. In human-computer interaction, expressive virtual assistants can improve user satisfaction and engagement. In educational tools, accurate facial animations are essential for training and learning.

In recent years, a significant shift in the animation industry has been made towards automation and efficiency. One promising approach that has emerged is text-based animation. This method leverages the richness of human languages through Natural Language Processing (NLP) and combines it with the incredible potential of AI to interpret textual descriptions and generate corresponding animations. By addressing the existing problems of cost and speed, this animation process has the potential to democratize the animation process, making it accessible to a broader audience and enabling new creative possibilities.

In the field of full-body motion synthesis, many previous works have shown promising high-quality results when tasked with text-to-motion or action-to-motion like [1] [2] [3] [6]. Recent works in this domain have extended from pure text or action-based motion generation to motion editing, smoothing, and styling. However, directly extending these methods to animate human heads cannot generate realistic animations, as we show in our results.

In the field of facial motion synthesis, previous works mainly focused on obtaining lip synchronous facial motions from a provided audio description [7] [8] [9] [10]. These projects showed good results in achieving lip synchronicity and enabling a finely-grained speaking

style. Nevertheless, these methods can not perform animation with user-friendly text prompts. In this thesis, we aim to tackle this particular challenge.

One major obstacle in this field is the lack of a large, high-quality (text, motion) dataset on which these models can be trained. This absence of a robust dataset presents a significant challenge to advancing research in this area. To alleviate this issue, we created the first large-scale (text, motion) dataset for the facial domain by combining the richness of in-the-wild facial (text, video) datasets [11] with the outstanding performance of 3D trackers [12]. To increase the overall quality of our dataset, we propose several properties that monocular input videos must uphold to be considered of good enough quality.

Using the created dataset, we propose the first sequential, transformer-based, text-to-motion model capable of generating realistic-looking facial motions from most textual descriptions. We propose leveraging the good performance of the pre-trained Language-to-Image model CLIP and its large amount of (text, image) training data to improve the performance of our model, not only due to the good performance of CLIP mapping language and visual information to the same space but also when handling unseen text prompts. We propose utilizing a conditional Vector Quantized Variational Autoencoder to increase the diversity of generated output sequences from same-sentence prompts and to provide guidance to the whole generated motion sequence. We generate the motion sequence by employing a transformer-based Decoder with applied Periodic Positional Encoding, Biased Causal Multi-Head Self-Attention and Biased Multi-Head Cross-Attention using the guidance of the cVQVAE.

We plan to make our dataset, the proposed pipeline, and our model publicly available for research purposes.

Overall, the key contributions of this thesis can be summarized as follows:

1. Propose the first large-scale dataset with (text, motion) pairs created directly from in-the-wild monocular videos
2. The first vector-quantized transformer-based sequence model that performs text-guided animation of human head avatars.
3. Compare our methods against existing state-of-the-art methods and achieve significantly better results both qualitatively and quantitatively

2 Thesis Overview

In this chapter, we briefly discuss the organization of this thesis.

In the next chapter, Chapter 3, we will summarize the fundamental concepts, models, and architectures we used throughout this work. We will start by introducing different representation formats for 3D morphable models. Then, we will provide an overview of the Language-to-Image models and CLIP specifically. Next, we will describe the two concepts of generative models used in this work, Variational Autoencoders (VAEs) and Vector Quantized Autoencoders (VQVAEs). Finally, we will recapture the transformer architecture and provide some insights into the more recent changes of its Positional Encoding.

In Chapter 4, we will provide an overview of the directly correlated works, datasets, and tools we used throughout this work. We will start by describing two datasets with a structure similar to our needs. Then, we will provide insights into two great tools for 3D tracking and reconstructing 3D morphable models from monocular images. Finally, we will provide an overview of existing projects aimed at generating motion sequences from textual descriptions in different domains.

We focus on our proposed dataset in Chapter 5. We start by defining selection criteria that our data must uphold and providing a reason for them. Afterward, we explain the steps of our pipeline in great detail. After that, we will define the limitations of the resulting dataset and, if possible, our solution. Finally, we will review some efficiency-improving measures undertaken as we needed to process a large amount of data.

After that, in Chapter 6, we will define our proposed model structure. To this end, we will start by defining the exact types of input our model requires. Next, we will present our main Encoder and Decoder blocks separately before defining the losses used to train the model and give some insights into why we decided to use them.

In Chapter 7, we will present the results of our model architecture. We will start by defining the baseline experiment of our final model. Here, we will give detailed information on the exact implementation used. Then, we will evaluate our baseline experiment both visually and quantitatively and provide a comparison to some ablation studies

Finally, in Chapter 8, we will reveal the limitations of our model architecture, provide some ideas on how we can improve upon it, and then in Chapter 9 we will give some concluding remarks about this thesis.

3 Background

This chapter will provide information on methods, structures, and ideas used throughout this work. We will first describe standard methods for 3D morphable models used to model head avatars. Then, we will provide an overview of the common language models we experimented with in this work. Afterward, we will discuss some commonly used generative network models when tasked with a textual condition. Finally, we will summarize the Transformer network type and its attention mechanism.

3.1 3D Morphable Face Models

A 3D Morphable Face Model (3DMM) [13] aims to disentangle the geometry of a face from its texture. It is constructed by statistically analyzing a set of 3D facial scans aligned to a reference frame, ensuring a consistent representation of both geometry and texture. On the obtained representation, Principal Component Analysis (PCA) is then applied to reduce the dimensionality of the data while retaining the most significant variations in shape and texture.

When using a 3DMM, any new face can be represented by a linear combination of the mean face and the principal components, weighted by coefficients. The weighted principal components represent the shape and texture deviations from the mean face and can be interpreted as morphing parameters from the mean face to the new face.

A 3DMM can be used in a wide variety of tasks in the field of Computer Vision and Graphics. For 3D face reconstruction, a 3DMM can be used to align the reconstructed face to the 2D input by optimizing on the model coefficients. For face recognition, a 3DMM can enhance accuracy by disentangling the dynamic expression from the static shape of the face. In animation and editing, a 3DMM enables the creation of diverse facial expressions and morphing between identities, ensuring realism through statistical grounding.

3.1.1 Basel Face Model

The Basel Face Model (BFM) [15] aims to provide an easy-to-use and powerful 3D shape and texture model for faces. The authors improved upon the previously existing works by offering a higher shape and texture accuracy due to a better scanning device and fewer correspondence artifacts due to an improved registration algorithm. The model separates pose, lighting, imaging, and identity parameters, which enables face recognition across datasets by

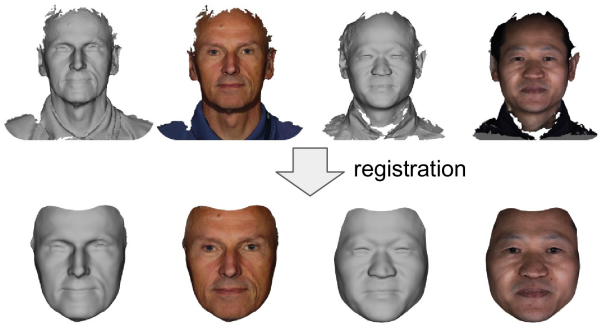


Figure 3.1: **Registration process:** This figure illustrates the necessity to register a 3D surface mesh and its 3D coordinates to a common set of vertices, where each vertex has a specific identity. This can be achieved by a registration process, mapping a canonical mesh template onto each raw 3D face image.[14]

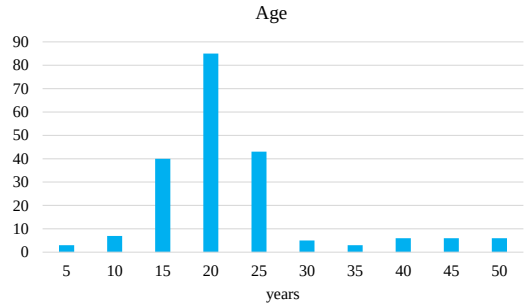


Figure 3.2: BFM Training Data - Age Distribution

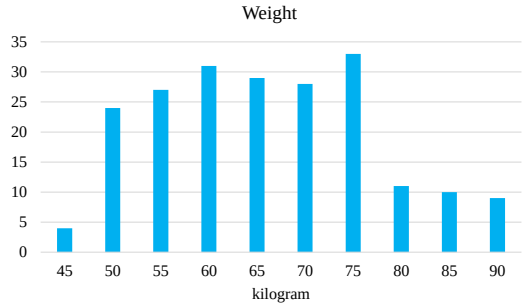


Figure 3.3: BFM Training Data - Weight Distribution

solely utilizing the identity parameter.

BFM was trained on a newly created dataset for which the authors sampled 100 female and 100 male subjects with varying ages and weights (see 3.2, 3.3). Each subject was scanned three times with neutral expressions, and the most natural-looking one was selected. The obtained 3D scans were then aligned by re-parametrizing such that semantically corresponding points (i.e., nose tips, eye corners, etc.) share the same position. As this re-parametrization procedure can result in a wrong registration, the authors employed the Optimal Step Nonrigid ICP Algorithm [16] to establish a better correspondence. This algorithm progressively deforms a base template towards the measured surface while ensuring a smooth deformation. Finally, the authors manually annotated landmarks on the lips, eyebrows, and ears. The final registered faces are represented as a triangular mesh with $m = 53490$ colored 3D vertices.

From the two $3m$ dimensional vectors for shape and texture, the BFM constructs two independent Linear Models that align a Gaussian Distribution to the data using PCA, resulting

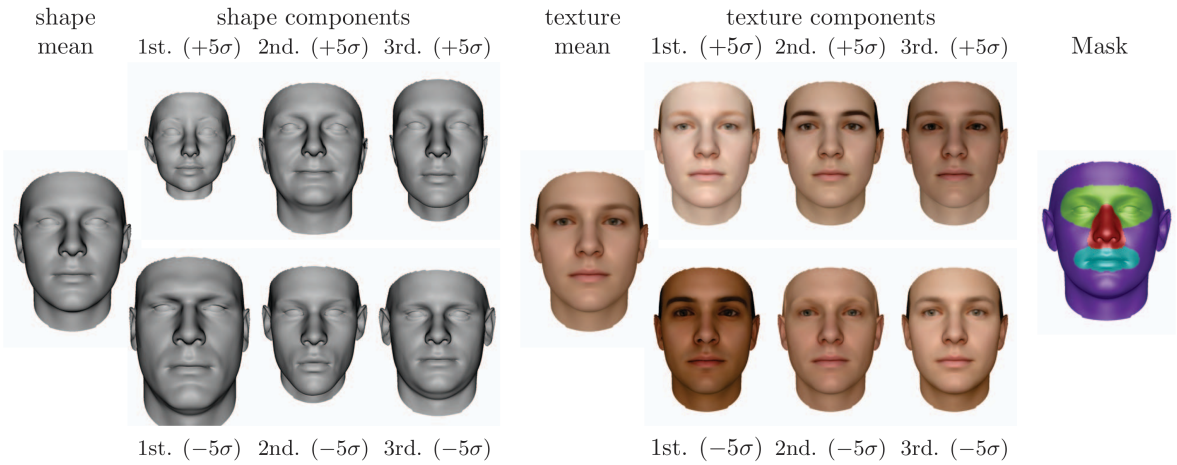


Figure 3.4: The mean together with the first three principal components of the shape (left) and texture (right) PCA model. Shown are the mean shape reps. texture plus/minus five standard deviations σ. Masks with the four manually chosen segments (eyes, nose, mouth, and rest) are used in the fitting to extend the flexibility. [15]

in the parametric face model consisting of

$$M_s = (\mu_s, \sigma_s, U_s) \quad \text{and} \quad M_t = (\mu_t, \sigma_t, U_t) \tag{3.1}$$

with μ denoting the mean, σ the standard deviation and U the orthonomal basis of the principal components, each for the shape and texture respectively.
The principal components are used as linear combinations to create new faces from the model.

$$s(\alpha) = \mu_s + U_s \operatorname{diag}(\sigma_s)\alpha \tag{3.2}$$
$$t(\beta) = \mu_t + U_t \operatorname{diag}(\sigma_t)\beta \tag{3.3}$$

All coefficients of the BFM are independent and normally distributed with unit variance under the assumption of normally distributed training examples and a correct mean estimation.

By its design, BFM does not model the entirety of a head, only its frontal parts. Especially movements of the neck cannot be modeled correctly. Besides this problem, BFM does not provide modeling for the eyes but only fills them with a generic blanc, as seen in 3.4.

3.1.2 FLAME

The FLAME (Faces Learned with an Articulated Model and Expression) [17] model provides an improvement upon the BFM 3DMM by not only disentangling the shape and texture but also the expression and pose.

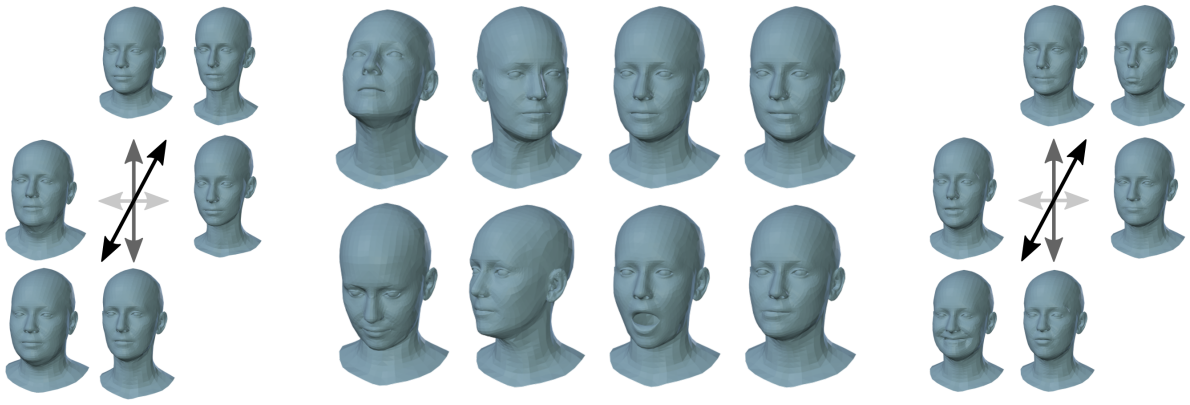


Figure 3.5: Parametrization of the FLAME model (female model shown). **Left**: Activation of the first three shape components between -3 and $+3$ standard deviations. **Middle**: Pose parameters actuating four of the six neck and jaw joints rotationally. **Right**: Activation of the first three expression components between -3 and $+3$ standard deviations.[17]

It adapts the SMPL model [18] formulation to the head space. The original SMPL body model does not model facial pose (articulation of jaw or eyes) or facial expression but exhibited high efficiency and compatibility with existing game engines.
In the original SMPL model, geometric deformations are caused by either the intrinsic shape of the subject or by deformations related to pose changes in the kinematic tree. When considering faces, the authors of FLAME denote that many deformations are due to muscular activation, which is unrelated to any articulated pose change. They therefore extend the SMPL model by incorporating an additional expression blend shape (see 3.5).

"FLAME uses standard vertex-based linear blend skinning (LBS) with corrective blendshapes, with $N = 5023$, $K = 4$ joints (neck, jaw, and eyeballs as shown in 3.6) and blendshapes, which will be learned from data."[17]

Overall, the FLAME model is described by the function:

$$M(\vec{\beta}, \vec{\theta}, \vec{\psi}) : \mathbb{R}^{|\vec{\beta}|\times|\vec{\theta}|\times|\vec{\psi}|} \rightarrow \mathbb{R}^{3N} \tag{3.4}$$

which takes coefficients describing shape $\vec{\beta} \in \mathbb{R}^{|\vec{\beta}|}$, pose $\vec{\theta} \in \mathbb{R}^{|\vec{\theta}|}$ and expression $\vec{\psi} \in \mathbb{R}^{|\vec{\psi}|}$.
"The model consists of a template mesh $\overline{\mathbf{T}} \in \mathbb{R}^{3N}$, in the 'zero pose' $\vec{\theta^*}$, a shape blend shape function, $B_S(\vec{\beta}; \mathcal{S}) : \mathbb{R}^{|\vec{\beta}|} \rightarrow \mathbb{R}^{3N}$, to account for identity-related shape variations, corrective pose blend shapes $B_P(\vec{\theta}, \mathcal{P}) : \mathbb{R}^{\vec{\theta}} \rightarrow \mathbb{R}^{3N}$, to correct pose deformations that LBS cannot solely explain, and expression blend shapes, $B_E(\vec{\psi}; \mathcal{E}) : \mathbb{R}^{\vec{\psi}} \rightarrow \mathbb{R}^{3N}$, that capture facial

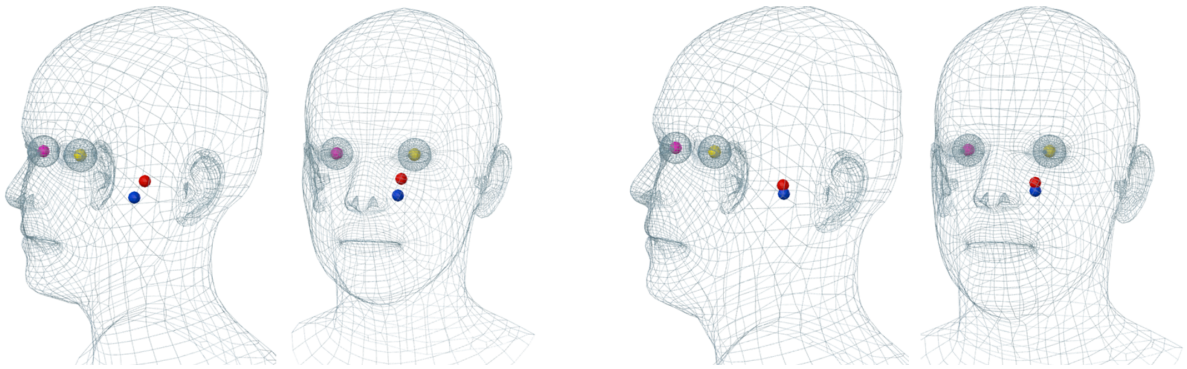


Figure 3.6: Joint locations of the female (left) and male (right) FLAME models.
Pink/yellow represents right/left eyes, red is the neck joint, and blue is the jaw.[17]

expressions. A standard skinning function $W(\overline{\mathbf{T}}, \mathbf{J}, \vec{\theta}, \boldsymbol{\mathcal{W}})$ is applied to rotate the vertices of $\overline{\mathbf{T}}$ around the joints $\mathbf{J} \in \mathbb{R}^{3K}$, linearly smoothed by blendweights $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{K \times N}$."[17]
All in all, the model is defined as

$$M(\vec{\beta}, \vec{\theta}, \vec{\psi}) = W(T_P(\vec{\beta}, \vec{\theta}, \vec{\psi}), \boldsymbol{J}(\vec{\beta}), \vec{\theta}\boldsymbol{\mathcal{W}}) \tag{3.5}$$

where

$$T_P(\vec{\beta}, \vec{\theta}, \vec{\psi}) = \overline{\boldsymbol{T}} + B_S(\vec{\beta}; \mathcal{S}) + B_S(\vec{\theta}; \mathcal{P}) + B_E(\vec{\psi}; \mathcal{E}) \tag{3.6}$$

denotes the template with shape, pose, and expression offsets.

3.2 Image-Text Models

Image-text models aim to match images and textual descriptions onto a shared latent space. Such models can be used in various Computer Vision tasks like image classification and annotation as they can provide the richness of natural language models to the visual domain.

The most prominent Image-Text model is the CLIP (Contrastive Language-Image Pre-Training) [19] model, presented in 2021. CLIP has been trained on a newly created dataset as existing text-to-image datasets [20] [21] [22] did not provide sufficiently large quantities of data after cleaning. The created dataset contained 400 million (image, text) pairs from various publicly available sources on the internet. To cover as broad a set of visual concepts as possible, the authors searched 500,000 queries containing all words occurring at least 100 times in the English version of Wikipedia. The resulting dataset has a total word count similar to the WebText dataset used for many Large Language Models.

As the computational complexity of this problem renders such a model almost impossible, the authors decided to utilize a contrastive pre-training scheme, providing a much-needed

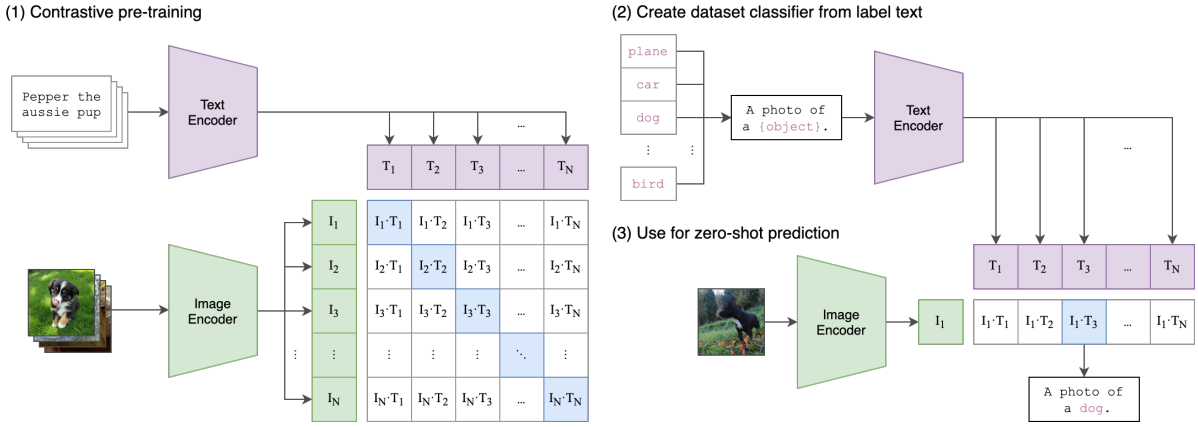


Figure 3.7: **Summary of the CLIP approach:** While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time, the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes [19]

simplification to the later regular training. In the pre-training scheme, the model is not trained to predict the *exact words of a text* but instead the *text as a whole*: "Given a batch of N (image, text) pairs, CLIP is trained to predict which of the $N \times N$ possible (image, text) pairings across a batch actually occurred. To do this, CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairs"[19].

The CLIP model is split into two parts: the Text-Encoder and the Image-Encoder. The Text-Encoder comprises a 12-layered Transformer with a 512-dimensional latent space and eight attention heads. It operates on a lower-cased byte pair encoding (BPE) text representation with a 49,152 vocabulary size. The authors decided to cap the textual sequence length at 77 tokens to improve the computational efficiency. For the Image-Encoder, the authors provide two versions with different architectures. The first architecture uses the popular ResNet-50 [23] with some modifications [24] [25] as basis. The second architecture uses the Vision Transformer (ViT) [26], again with minor modifications, as basis. The ViT-based architecture yielded better results, but Radford et al. provided both models.

3.3 Generative Models

Two key difficulties can be identified when using AI models to generate novel data. The first can be described by the fidelity of the output to the condition, meaning how well the model can be conditioned. The second can be described by the variability in the output, meaning

how diverse the model output is.
Several training approaches or model structures have been developed to tackle these goals. The most common ones are Generative Adversarial Networks (GANs), Diffusion-based models, or Variational Autoencoders (VAEs) with their special case, the Vector Quantized Variational Autoencoder (VQVAEs) are used.
This section will provide an overview of the latter two, VAEs and VQVAEs, used in this work.

3.3.1 Variational Autoencoders

A Variational Autoencoder is a generative model based on stochastic likelihood. It "can be viewed as two coupled, but independently parametrized models: the encoder or recognition model, and the Decoder or generative model. These two models support each other. The recognition model delivers to the generative model an approximation to its posterior over latent random variables, which it needs to update its parameters inside an iteration of "expectation maximization" learning. Reversely, the generative model is a scaffolding for the recognition model to learn meaningful representations of the data, including possibly class labels. The recognition is the approximate inverse of the generative model according to Bayes Rule" [27].

A "VAE learns the stochastic mappings between an observed $\boldsymbol{x}$-space, whose empirical distributions $q_D(x)$ is typically complicated, and a latent $\boldsymbol{z}$-space, whose distribution can be relatively simple. The generative model learns a joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ that is often (but not always) factorized as $p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$, with a prior distribution over latent space $p_\theta(\boldsymbol{z})$, and a stochastic Decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. The stochastic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, also called *inference model*, approximates the true, but intractable posterior $p_\theta(\boldsymbol{z}|x)$ of the generative model"[27].
This process can be described in Figure 3.8 [27].

"The optimization objective for the Variational Autoencoder, like in other variational methods, is the *evidence lower bound*, abbreviated as ELBO"[27]. The ELBO can be derived by the following equation:

$$\log p_\theta(\boldsymbol{x}) = \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]\right]}_{=\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) \quad \text{(ELBO)}} + \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right]}_{=D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x}))} \tag{3.7}$$

Here, the first part describes the ELBO objective while the second one describes the Kullback-Leibler (KL) divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.
Therefore the optimization of ELBO ($\mathcal{L}_{\theta,\phi}(\boldsymbol{x})$) w.r.t. the parameters θ and ϕ, will optimize two goals [27]:

1. It will approximately maximize the marginal likelihood $p_\theta(\boldsymbol{x}$.
   This means our model will become better.

2. It wil minimize the KL divergence of the approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ from the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, so $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ becomes better.

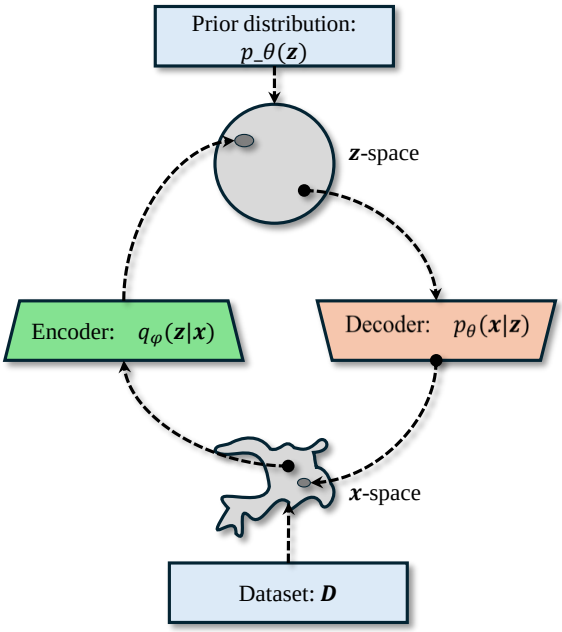


Figure 3.8: **VAE Stochastic Goal Formulation [27]:** The VAE learns a stochastic mapping between the observed $\boldsymbol{x}$-space, whose distribution $q_D(\boldsymbol{x})$ is typically complicated, and a latent $\boldsymbol{z}$-space, whose distribution can be relatively simple. The generative model learns a joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with $p_\theta(\boldsymbol{z})$ as prior distribution over the latent space and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ as stochastic decoder. The stochastic encoder $q_\phi(\boldsymbol{z}|x)$ approximates the true, but intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ of the generative model.

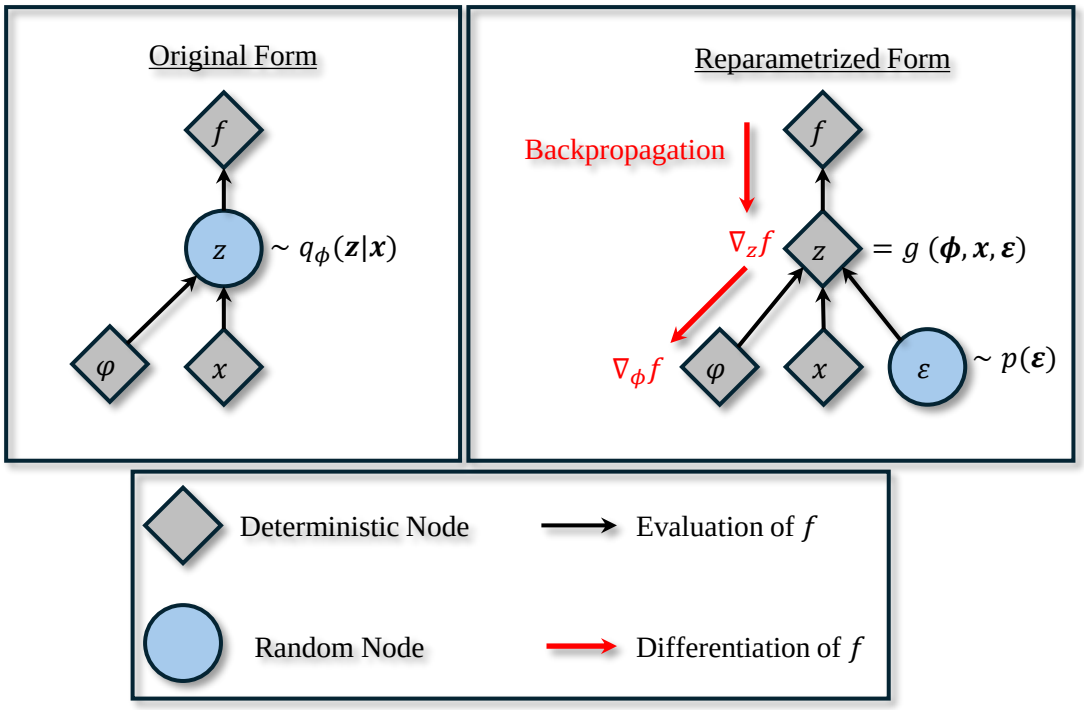


Figure 3.9: **Illustration of the reparameterization trick:** The variational parameters ϕ affect the objective f through the random variable $z \sim q_\phi(z|x)$. We wish to compute gradients $\nabla_\phi f$ to optimize the objective with SGD. In the original form (**left**), we cannot differentiate f w.r.t. ϕ because we cannot directly backpropagate gradients through the random variable z. We can 'externalize' the randomness in z by re-parametrerizing the variable as a deterministic and differentiable function of ϕ, x and a newly introduced random variable ε. This allows us to 'backprob through z', and compute gradients $\nabla_\phi f$. [27]

To optimize a Variational Autoencoder, the so-called "Reparamatrization Trick" is usually used to optimize a Variational Autoencoder. It ensures that for continuous latent variables, the ELBO can be straightforwardly differentiated w.r.t. both ϕ and θ through a change of variables:

$$z = g(\varepsilon, \phi, x) \tag{3.8}$$

with the distribution of the latent variable ε is independent from x or ϕ

Usually, a VAE is implemented as described in Figure 3.10 and contains two losses, the Reconstruction loss and the KL-Divergence loss derived earlier.

By default, a VAE cannot be conditioned by the generated output, as it "assumes" all data to be from the same distribution. As often seen in generative tasks, if one wants to create outputs conditioned through some measure (i.e., text, audio, etc.), we must provide conditioning

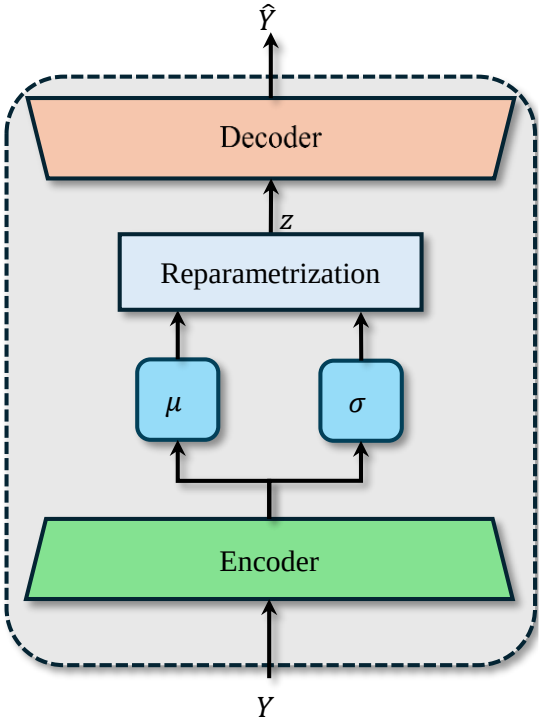


Figure 3.10: **Variational Autoencoder:** The Encoder encodes the input into a mean μ and standard deviation σ value. Following the reparametrization trick, these values are used to sample from the predicted distribution and provided to the Decoder.

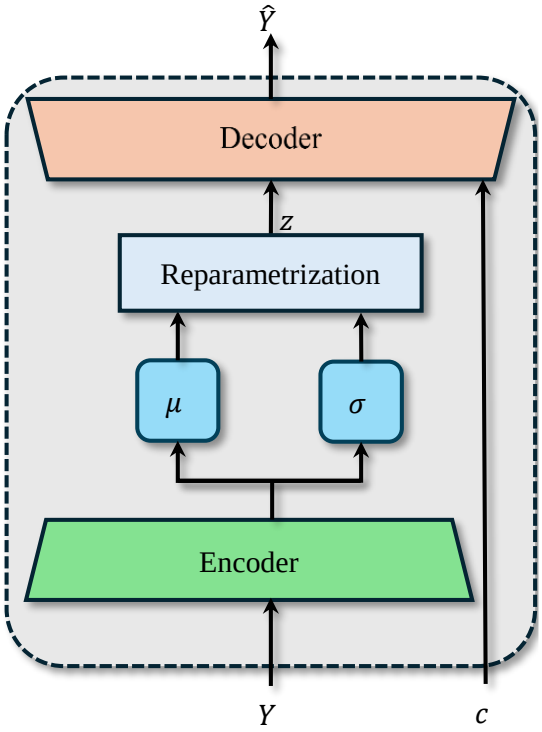


Figure 3.11: **Conditional Variational Autoencoder:** In addition to a sampled value from the predicted distribution, a conditioning code c is provided as input.

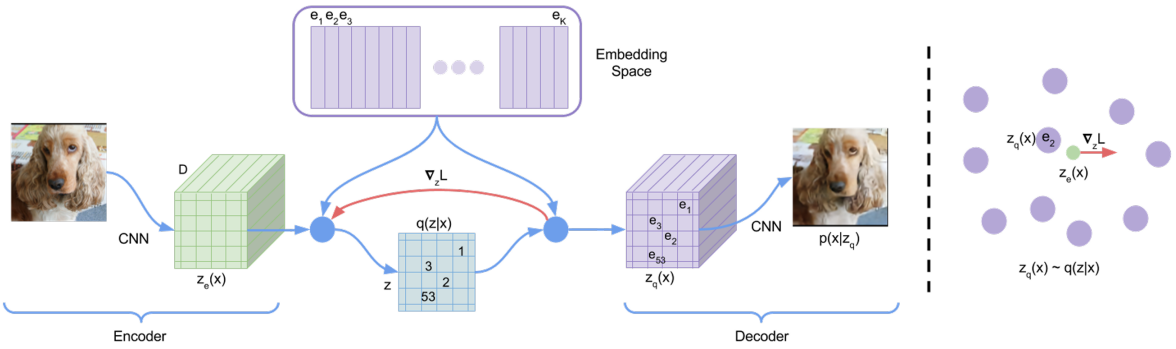


Figure 3.12: **Left**: A figure describing the VQVAE. **Right**: Visualization of the embedding space. The Encoder $z(x)$ output is mapped to the nearest point e_2. The gradient $\nabla_z \mathcal{L}$ (in red) will push the Encoder to change its output, which could alter the configuration in the next forward pass.

information c to the VAE. This results in the Conditional Variational Autoencoder (cVAE) seen in Figure 3.11

3.3.2 Vector Quantized Variational Autoencoders

A Vector Quantized Variational Autoencoder (VQVAE)[28] takes a different but similar approach to the VAE. In contrast to the VAE, the VQVAE produces discrete, rather than continuous, codes, and the prior is learned rather than static. The discretization of the latent space is achieved by incorporating ideas of Vector Quantization (VQ). This "allows the model to circumvent issues of 'posterior collapse' - where latent are ignored when they are paired with a powerful autoregressive Decoder - typically observed in the VAE framework" [28].

Van den Oord et al. [28] ensure discrete latent variables by defining a latent space $e \in \mathbb{R}^{K \times D}$ with K denoting the size of the latent space and D denoting the dimensionality of each latent embedding vector e_i. The discrete latent variable z is then calculated by the nearest neighbor look-up using the shared embedding space as the so-called "Codebook". The input to the Decoder is this corresponding nearest neighbor e_k. The "posterior categorical distribution $q(z|x)$ probabilities are defined as one-hot as follows:

$$q(z = k|x) = \begin{cases} 1 & \text{for } k = \text{argmin}_j ||z_e(x) - e_j||_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Where $z_e(x)$ is the output of the encoder network. We view this model as a VAE in which we can bound $\log p(x)$ with the ELBO"[28]. The proposed distribution $q(z = k|x)$ is deterministic, and by defining a simple uniform prior over z, they can obtain a KL divergence constant equal to $\log K$.

A VQVAE is trained by extracting the nearest embedding $z_q(x)$ from the Codebook, which is then passed to the Decoder during the forward pass of the SGD (Stochastic Gradient

Descent) algorithm. During the backward pass, the gradient $\nabla_z \mathcal{L}$ is passed unaltered to the Encoder, changing the Encoder output during the next forward pass. The total loss function can thus be described as follows:

$$\mathcal{L} = \log p(x|z_q(x)) + ||\text{sg}\left[z_e(x)\right] - e||_2^2 + \beta||z_e(x) - \text{sg}[e]||_2^2 \tag{3.10}$$

Here, sg stands for the stop-gradient operator, defined as the identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant. The first term is the reconstruction loss, which optimizes the Encoder and Decoder. The second term corresponds to the Vector Quantization dictionary learning term. It updates the Codebook by calculating the L2 loss to move the embedding vectors e_i towards the Encoder outputs $z_e(x)$. The third term represents a commitment loss, ensuring that the Encoder commits to an embedding and thus enforcing the embedding space to remain within certain bounds.

3.4 Transformers

Since the publication in 2017 by Vaswani et al. [29], the Transformer architecture has revolutionized the field of AI architectures. Nowadays, it is used in many of the best-performing works. But the core idea of Transformers remains fairly simple. Instead of combining Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) with an attention mechanism, Vaswani et al. decided to base every part of the Transformer on it.
In this section, we will first provide an overview of the usual structure of a Transformer, and then we will move to the attention mechanism. Finally, we will present several different types of Positional Encodings that have been proposed since the first inception of the Transformer architecture.

3.4.1 General Structure

A Transformer is typically split into an Encoder and a Decoder part, but both structures are widely used independently. Encoder and Decoder each consist of multiple (in the paper, the authors used $N = 6$ internally identical, but different between Encoder and Decoder, layers.

In the Encoder, each layer comprises a multi-head self-attention block, followed by a position-wise fully connected feed-forward network. After both the self-attention block and the feed-forward network, a residual connection and layer normalization are used, enabling a better gradient flow while preventing exploding values.

In contrast to the Encoder, the Decoder is provided with the shifted sequential output. To maintain the causality in the sequence, the Decoder contains a Masked Multi-Head Self-Attention block instead of an unmasked one. The next block consists of a Multi-Head Cross-Attention with only the query matrix from the previous block. The key and value

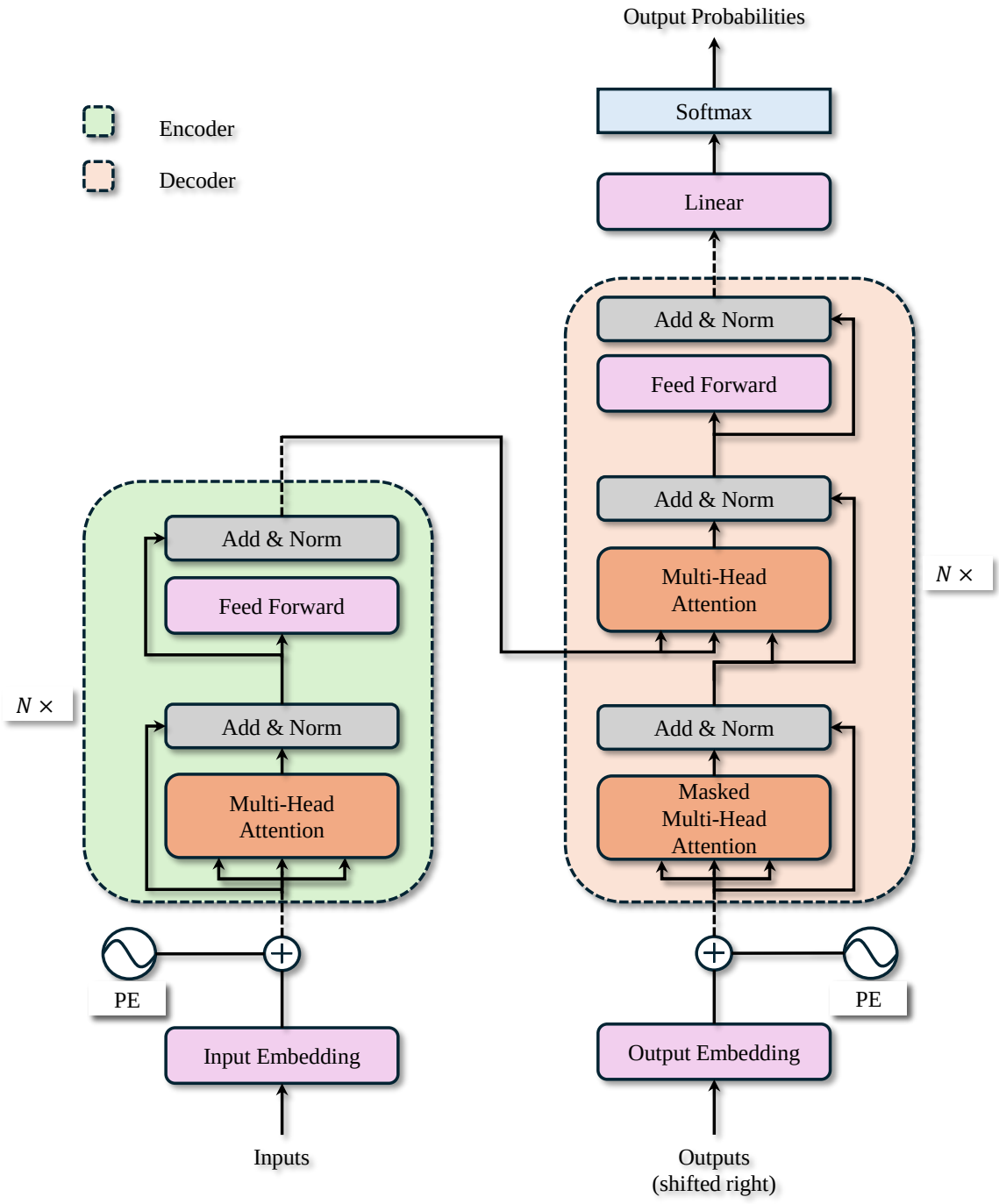


Figure 3.13: **Transformer Overview:** This figure illustrates the Transformer architecture as proposed in [29]. **Left** describes one Encoder Layer with its two main blocks, the Multi-Head Self-Attention and a Feed-Forward Network. **Right** describes one Decoder Layer with its three main blocks: a Masked Multi-Head Self-Attention ensuring causality, a Multi-Head Cross-Attention with the memory from the Encoder, and the Feed-Forward Network. All blocks are followed by adding the residual connection and the normalization. Encoder and Decoder Layers are repeated multiple (in [29]: $N = 6$) times.

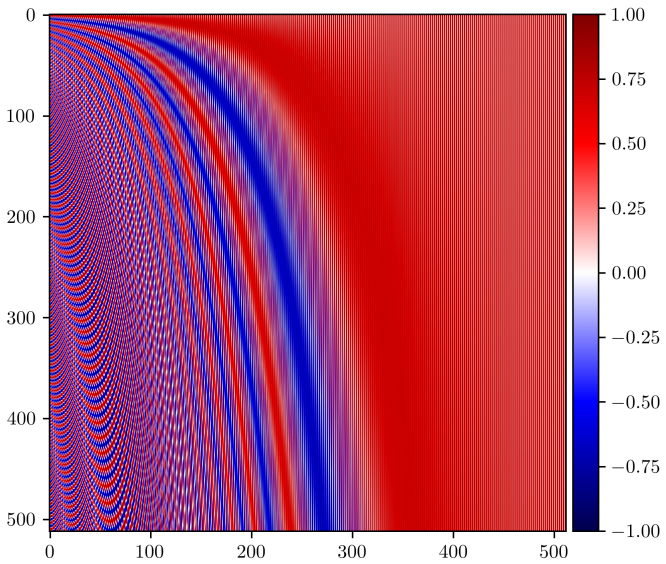


Figure 3.14: **Sinusoidal Positional Encoding:** This figure illustrates the Sinusoidal Positional Encoding. Here the x-axis represents the dimension of the model d_{model} and the y-axis the position pos in the sequence.

matrices originate from the Encoder. As the final block, the Decoder also contains a position-wise fully connected feed-forward network. After every block, the Decoder also uses layer normalizations with a residual connection following the same reasoning as the Encoder.

3.4.2 Positional Encoding

Before any data is provided into the Transformer, a Positional Encoding (PE) is applied. This is done as the model does not contain any recurrence or convolution, resulting in a lack of information about the sequential ordering. Positional Encoding alleviates this problem by usually adding unique information about the position of each element in the sequence to the elements themselves.
In the original work by Vaswani et al. [29], a Sinusoidal Positional Encoding was used, but since then, several different versions of Positional Encoding have been developed and are in wide use today.

Sinusoidal Positional Encoding

Sinusoidal Positional Encoding can be described by the following equation, describing the unique information added to each element of the sequence:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{3.11}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{3.12}$$

Periodic Positional Encoding

The Periodic Positional Encoding was introduced in the FaceFormer project [7]. In the FaceFormer project, Fan et al. used Attention with Linear Biases [30] (ALiBi). Still, they noticed that it does not provide any temporal information directly, resulting in a static output. When incorporating the default Sinusoidal Positional Encoding, they observed minimal generalization ability for longer sequences.
Therefore, they devised the Periodic Positional Encoding for injecting temporal order information while maintaining compatibility with ALiBi:

$$\mathrm{PPE}_{t,2i} = \sin\left((t \mod p)/10000^{2i/d}\right) \tag{3.13}$$

$$\mathrm{PPE}_{t,2i+1} = \cos\left((t \mod p)/10000^{2i/d}\right) \tag{3.14}$$

Rotary Positional Encoding

The Rotary Positional Encoding (RoPE) was first introduced in the RoFormer project [31].

With RoPE, the authors aim to leverage both previously existing fields of Positional Encoding: Absolute Positional Encoding [7] [29] and Relative Positional Encoding [32].
With the regular Sinusoidal Positional Encoding and PPE belonging to the first category.
Relative Positional Encoding aims to incorporate the positional information not to the input of the Transformer but to the key and value matrices.

For the RoPE, the authors defined their goal to find an encoding mechanism that solves the functions $f_q(x_m, m)$ and $f_k(x_n, n)$ and confirms the following relation:

$$\langle f_q(x_m, m), f_k(x_n, n)\rangle = g(x_m, x_n, m-n) \tag{3.15}$$

2D-Case: In the 2D-Case, the authors provide a simple solution as a proof of concept:

$$f_x(x_m, m) = (W_q x_m)e^{im\theta} f_k(x_n, n) = (W_k x_n)e^{in\theta} g(x_m, x_n, m-n) = \mathrm{RE}[(W_q x_m)(W_k x_n)^* e^{i(m-n)\theta}] \tag{3.16}$$

with $\mathrm{RE}(\dot{})$ denoting the real part of a complex number and $(W_k x_n)^*$ representing the conjugate complex number of $(W_k x_n)$. $\theta \in \mathbb{R}$ is a preset non-zero constant.
The authors also provide a more intuitive matrix formulation:

$$f_{\{q,k\}}(x_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & cosm\theta \end{pmatrix} \begin{pmatrix} W^{(1,1)}_{\{q,k\}} & W^{(1,2)}_{\{q,k\}} \\ W^{(2,1)}_{\{q,k\}} & W^{(2,2)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix} \tag{3.17}$$

We refer to the original work by Su et al. [31] for the general version.

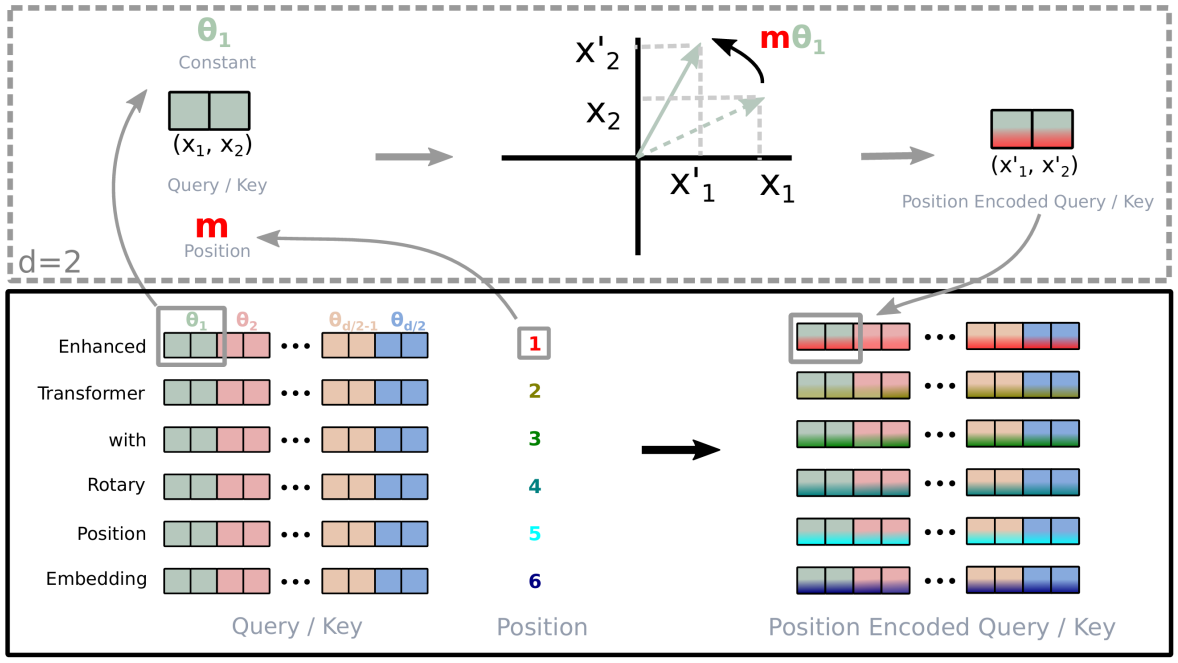


Figure 3.15: **Implementation of Rotary Positional Encoding [31]:** This figure illustrates the rotation of the Query and Key matrices by a constant value θ and the value m representing the position of the element in the sequence.

3.4.3 Attention Mechanism

Every Attention Block follows the core idea of the attention function with only minor variations in the exact implementation. The "attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key"[29].

In the Transformer Architecture, Vaswani et al. decided to use the Dot-Product (multiplicative) attention function instead of the additive attention for performance reasons, as the multiplicative attention can be computed far quicker and in parallel using the highly optimized matrix-multiplication code. Vaswani et al. describe that in practice, the attention function is computed on a set of queries simultaneously, packed into a matrix Q. At the same time, the keys and values are also packed into matrices K and V, respectively. Therefore, the matrix output can be computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.18}$$

Here, the authors differ from the previously commonly used attention by including the scaling factor, hence Scaled Dot-Product Attention, $\frac{1}{\sqrt{d_k}}$.

If the query, keys, and values originate from the same previous block, the Attention Block is called Self-Attention, as it is "attending" to itself. In Cross-Attention, the key and value matrices originate from the Transformer Encoder, while the query originates from the previous step in the Transformer Decoder. If a mask prevents the model from attending to specific elements, the Attention Block is called Masked (Self-)Attention. Such a mask can be used to ensure causality in the Decoder, where a triangular mask prevents the Attention Block from attending to any sequence elements before the current one.

In Multi-Head Attention, the K, Q and V matrices are split into smaller dimensions, d_k, d_k and d_v respectively, using linear projections. The attention function is calculated parallelly on each projected matrix, yielding d_v-dimensional output values. They are then concatenated and once again projected, resulting in the final values.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \ldots \text{head}_h)W^O \tag{3.19}$$

$$\text{with head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.20}$$

Here the projections are parameter matrices of shapes: $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

4 Related Work

This chapter will summarize existing literature and projects analyzed and used during this thesis. We will split this chapter into four parts, each introducing one important aspect we used.

4.1 Dataset

As with all AI-related tasks, the dataset is of detrimental importance. Especially the quality and quantity of the samples have a substantial impact on the resulting performance of the model. This section will provide an overview of existing datasets focusing on facial motion.

4.1.1 CelebV-Text

The CelebV-Text dataset [11] was presented at the CVPR 2023 and consists of 70,000 in-the-wild facial video clips covering various motions. For each video, the authors provided 20 texts generated in a semi-automatic manner using a probabilistic context-free grammar (PCFG) from manually assigned attributes to timesteps in the videos, which were also provided.

The authors collected the videos using the same strategy they had employed during the construction of the CelebV-HQ [33]. Specifically, they first generated a large number of queries, including names, movie titles, vlogs, etc., to retrieve videos containing human faces with temporally dynamic state changes and abundant facial attributes [11]. With these queries, the authors collected and downloaded the videos from online sources (i.e., Youtube) while filtering videos with a too-short duration (< 5s) and low resolutions ($< 512 \times 512$), which initially were part of the CelebV-HQ dataset. To handle videos with a larger resolution, the authors opted to cut out the facial bounding boxes rather than resize them to a fixed size to maintain the original video quality. The resolution distribution in the resulting dataset can be described by Table 4.1.

Resolution	Percentage
$512^2 \sim 1024^2$	56.4%
$> 1024^2$	43.6%

Table 4.1: Resolution Distribution

(a) Video Samples (left: general appearance, middle: detailed appearance, right: light conditions) (b) Text Descriptions

Figure 4.1: **Overview of CelebV-Text:** CelebV-Text contains (a) 70,000 video samples and (b) 1,400,000 text descriptions. Each video sample is annotated with general appearance, detailed appearance, light conditions, action, emotion, and light direction[11].

With the goal of reducing face area noise, the authors decided to split a video into different clips when the background changes by a too large degree. For this goal, they utilized a toolkit [34].

The thus obtained video clips were then manually assigned attributes describing the information in the video. These attributes are split into two main categories: static attributes, describing the appearance of the person and the fixed lighting conditions, and dynamic attributes, describing the action, emotion, and lighting directions.

These attributes are then used as input for the PCFG to generate 20 textual descriptions. The rules of the PCFG can be described by Table 4.3, but we also refer to the original implementation and work for further details.

In the paper, the authors provide a lot of information about the distributions in the dataset. The dataset consists of predominately White (> 50%) persons, roughly 25% of the persons being of Asian origin, and people of Indian heritage being significantly underrepresented (see 4.2). Most persons are 25 to 35 years old, with only small outliers with age above 50 or below 25 (see 4.3). The clips provided are mostly (> 55%) of a length between 5 and 10 seconds, roughly 20% being a length of 10 to 15 seconds, and the number of videos gradually declining with increasing length (see 4.4). Almost all (> 80%) of the video clips contained some form of talking, with none of the other attributes having any similar presence in the dataset.

Overall, the CelebV dataset provides a valuable collection of in-the-wild, annotated video clips. But it exhibits two main drawbacks:

1. While each frame of the clips can have multiple attributes assigned to them, the textual

Static Attributes					
(a) General Appearance					
blurry	male	young	chubby	pale skin	rosy cheeks
oval face	receding hairline	bald	bangs	black hair	blond hair
gray hair	brown hair	straight hair	wavy hair	attractive	arched eyebrows
bushy eyebrows	bags under eyes	eyeglasses	mouth slightly open	smiling	big nose
pointy nose	high cheeks	big lips	double chin	no beard	5 o'clock shadow
goatee	sideburns	mustache	heavy makeup	wearing earrings	wearing hat
wearing lipstick	wearing necklace	wearing necktie	narrow eyes		
(b) Detailed Appearance					
Mole	freckle	one eyed	scar	dimple	
(c) Light Conditions					
dark	normal	bright	warm white	cool white	daylight
Dynamic Attributes					
(a) Action					
blow	chew	close eyes	cough	cry	drink
eat	frown	gaze	glare	head wagging	kiss
laugh	listen to music	look around	make a face	nod	play instrument
read	shake head	shout	sigh	sing	sleep
smile	smoke	sneeze	sniff	sneer	talk
turn	weep	whisper	win	yawn	blink
squint					
(b) Emotion					
Neutral	Happy	Sad	Anger	Fear	Surprise
Contempt	Disgust				
(c) Light Directions					
front	left 45	right 45	left 90	right 90	back

Table 4.2: CelebV-Text contains both static and dynamic attributes, including 40 general appearances, 5 detailed appearances, 6 lighting conditions, 37 actions, 8 emotions, and 6 light directions [11]

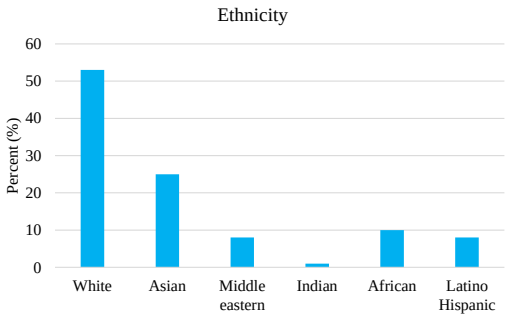


Figure 4.2: Ethnicity Distribution

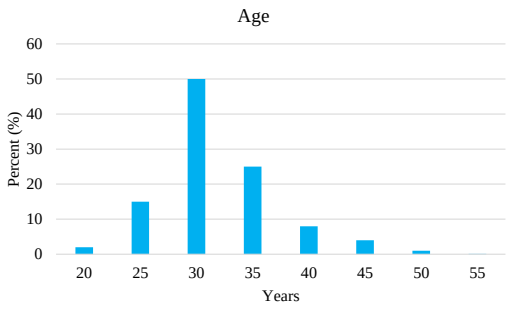


Figure 4.3: Age Distribution

Rule		Probability
S	⟶ NP VP	1.0
NP	⟶ Det Gender	0.5
NP	⟶ PN	0.5
VP	⟶ Wearing PN Are PN HaveWith	0.166
VP	⟶ Wearing PN HaveWith PN Are	0.166
VP	⟶ Are PN HaveWith PN Wearing	0.166
VP	⟶ HaveWith PN Are PN Wearing	0.166
VP	⟶ HaveWith PN Wearing PN Are	0.166
Wearing	⟶ WearVerb WearAttributes	1.0
Are	⟶ IsVerb IsAttributes	1.0
HaveWith	⟶ HaveVerb HaveAttributes	1.0
Det	⟶ **a**	0.333
Det	⟶ **the**	0.333
Det	⟶ **this**	0.333
Gender	⟶ **gender_related_attributes**	0.8
Gender	⟶ **Person**	0.2
PN	⟶ **personal_noun**	1.0
WearVerb	⟶ **is wearing**	0.5
WearVerb	⟶ **wears**	0.5
IsVerb	⟶ **is**	1.0
IsAttribute	⟶ **is_related_attributes**	1.0
HaveVerb	⟶ **has**	0.5
HaveVerb	⟶ **has got**	0.5
HaveAttributes	⟶ **has_related_attributes**	1.0

Table 4.3: Detailed PCFG design for generating descriptions for general faces [11]

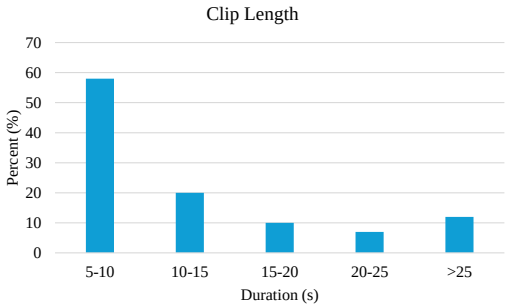


Figure 4.4: Length Distribution

Action Attribute	Distribution
talk	0.82
head wagging	0.23
turn	0.18
smile	0.15
gaze	0.11
blink	0.10

Table 4.4: Most common action attributes

descriptions often do not correctly reflect this. Instead, the descriptions sometimes would lable them as sequential, resulting in a mismatch between the description and the ground truth attributes.

2. Some of the used attributes, especially "talking", are not granulated finely enough. Instead, they combine almost all the different styles to talk into one attribute, resulting in an unnecessary decrease in descriptive accuracy.

4.1.2 FaMoS

The FaMoS (Facial Motion across Subjects) dataset [9] consists of 95 subjects performing 28 motion sequences each. Every sequence is described by one of six expressions (Anger, Disgust, Fear, Happiness, Sadness, and Surprise), two head rotations (left / right and up / down), and diverse facial motions, including extreme and asymmetric expressions. The data is captured using a multi-camera active stereo system with eight pairs of gray-scale stereo and color cameras, each calibrated, and the parameters for extrinsics, intrinsics, and radial distortions are measured. From the resulting 1600×1200 image sequences at 60 fps, the authors retrieved the 3D scans, each with about 110K vertices, using a multi-view stereo method. The resulting 3D mesh is then registered to the FLAME representation space is defined in the original FLAME [17] work by minimizing the vertex-to-vertex distance between the input mesh and the output of the FLAME model.

The dataset is provided in multiple formats, from the original images over the 110K vertex mesh to the $N = 5023$ vertices obtained from FLAME. The authors noted that the corresponding FLAME registration (β, θ, ψ) was not provided to maintain compatibility with different FLAME versions. However, obtaining the FLAME registrations from the vertex format is straightforward.

In the dataset, the following motions are provided for each of the 95 subjects:

Expressions: anger, disgust, fear, happiness, sadness, surprise

Facial Motions: bareteeth, blow_cheeks, checks_in, eyebrow, high_smile, jaw, kissing, lip_corners_down, lips_back, lips_up, mouth_down, mouth_extreme, mouth_middle, mouth_open, mouth_side, mouth_up, rolling_lips, sentence, smile_closed, wrinkle_nose

Head Rotation: head_rotation_left_right, head_rotation_up_down

The motion sequences have varying lengths from 150 to 450 frames. The average length of the sequence is 225 frames (3.5 seconds).

The FaMoS dataset provides a high-quality match between action classes and a registered 3D sequence. However, the dataset has three main problems:

1. It lacks in diversity of subjects, as it was only captured from 95 persons

Figure 4.5: **FaMoS Example Meshes:** This figure illustrates several poses of the FaMoS dataset from different actors with the goal of depicting a different expression.

2. The sequences are mostly very short, with an average sequence length of 225 frames
3. The data was captured with the explicit goal of depicting one specific motion, resulting in a lack of diversity compared to in-the-wild datasets.

4.2 3D Tracking

As none of the aforementioned datasets directly matched our expectations, we looked into ways to update the CelebV-Text [11] dataset to our needs as it matched our expectations the most.
To this end, we analyzed existing works that enable the generation of 3D models from monocular images and 3D motion sequences from monocular videos. This analysis yielded two prominent projects, DECA [35] and its successor MICA [12], which we will present in this section.

4.2.1 DECA

The DECA (Detailed Expression Capture and Animation)[35] project aims to generate 3D head avatars in the FLAME [17] representation from standard in-the-wild images. To this end, they trained a model "to robustly produce a UV displacement map from a low-dimensional latent representation that consists of person-specific detail parameters and generic expression parameters, while a regressor is trained to predict detail, shape, albedo, expression, pose and illumination parameters from a single image." [35].

The tracking process proposed by DECA is split into three steps. The first step aims to extract a coarse 3D reconstruction from the input monocular image in an Analysis-by-Synthesis way. The second step augments the coarse registration obtained by employing a detailed UV displacement map. The final step aims to disentangle the extracted details into static and dynamic ones.

To obtain the coarse 3D reconstruction, the input image $\boldsymbol{I}$ is passed through an Encoder E_c and the pre-trained FLAME model as a Decoder, which aims to reconstruct the image, rendered into 2D using a differentiable renderer. The Encoder is thus trained to extract the FLAME parameters β, ψ, θ in the latent space after the Encoder.
The coarse model is trained by minimizing the following loss function:

$$\mathcal{L}_{coarse} = \mathcal{L}_{lmk} + \mathcal{L}_{eye} + \mathcal{L}_{pho} + \mathcal{L}_{id} + \mathcal{L}_{sc} + \mathcal{L}_{reg} \tag{4.1}$$

Here, the **Landmark re-projection loss** ($\mathcal{L}_{lmk}$) measures the distance between the detected 2D ground-truth landmarks in the input image $\boldsymbol{I}$ and the corresponding 3D landmarks from the predicted reconstruction modified by the estimated camera model. The second loss term is defined as the **Eye closure loss** and aims to ensure the correct 3D modeling of the eyelids. The **Photometric Loss** ($\mathcal{L}_{pho}$) computes the error between the input Image I and the reconstruction output I_r and is computed by applying a binary face mask (V_I) to

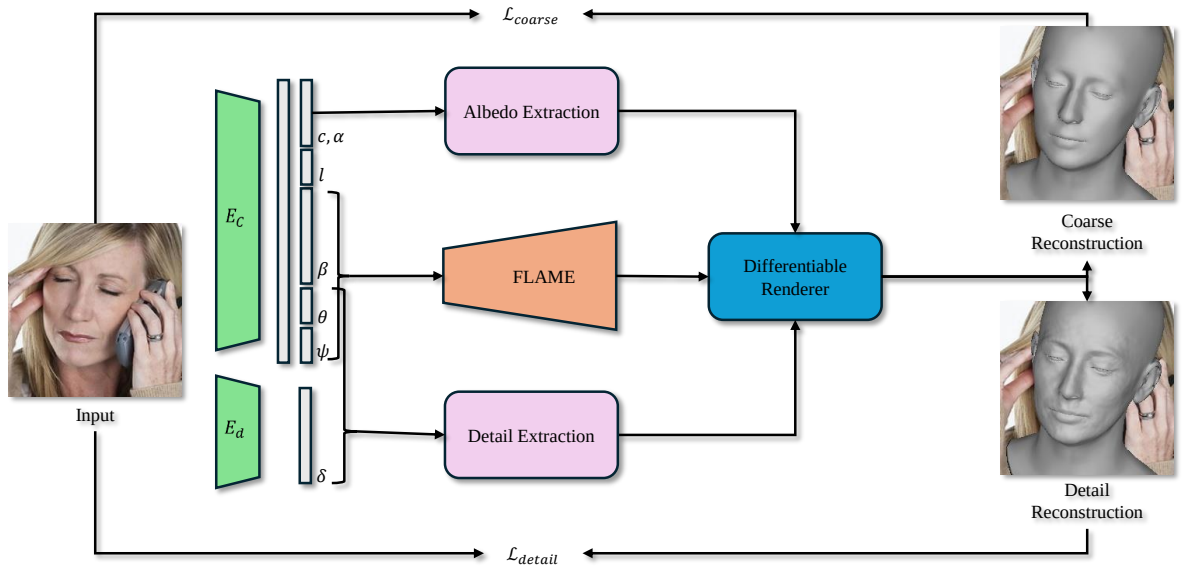


Figure 4.6: **DECA - Tracking Pipeline [35] (Abbreviated):** In DECA, the input image is encoded into parameters for the extraction of an albedo map, a detail map and the FLAME parameters β, θ, ψ which are used to obtain a 3D reconstruction. The 3D reconstruction is then differentiable rendered with obtained camera parameters to obtain a 2D reconstruction of which the reconstruction error to the original image is calculated.

the reconstruction error. The **Identity Loss** ($\mathcal{L}_{id}$) aims to ensure the reconstruction of more realistic face shapes by minimizing the difference of a face recognition network f applied to I and I_r. The second to last term defines the **Shape consistency loss** ($\mathcal{L}_{sc}$), which ensures the consistency of the FLAME shape parameters β between two input images I_i and I_j of the same person. As the usual shape consistency loss did not result in the desired effect, the authors redesigned it. Instead of enforcing the distance between the shape parameters β_i and β_j to be smaller than a certain margin, they propose a strategy in which β_i is replaced by β_j. As both are obtained from the same person, this optimizes the network to maintain consistency in the shape parameters between the two FLAME registrations. The final term represents the **Regularization** already utilized in the original FLAME [17] project. It regularizes the shape $E_\beta = ||\beta||_2^2$, expression $E_\psi = ||\psi||_2^2$ and albedo $E_\alpha = ||\alpha||_2^2$.

The detail reconstruction step augments the earlier attained coarse reconstruction with a detailed UV displacement map $D \in [-0.01, 0.01]^{d \times d}$. Here, the authors also train an Encoder E_d with the same architecture as E_c, but the latent space is represented by a 128-dimensional vector δ, representing subject-specific details. δ is then concatenated with the previously obtained FLAME parameters for the expression ψ and jaw pose θ_{jaw} and decoded by F_d to D

$$D = F_d(\delta, \psi, \theta_{jaw}) \tag{4.2}$$

The authors reasoned the use of the expression and jaw parameters due to their influence on the dynamic expression wrinkle details. The detail map D is then used to generate images with mid-frequency surface details. To reconstruct the detailed mesh M', the authors converted the geometric prior M and its surface normals N into the UV space and combined them with D:

$$M'_{uv} = M_{uv} + D \odot N_{uv} \tag{4.3}$$

Thus, the detail rendering I'_r can be obtained by rendering M with the normal map as:

$$I'_r = \mathcal{R}(M, B(\alpha, I, N'), c) \tag{4.4}$$

This second part of the model is trained by minimizing:

$$\mathcal{L}_{detail} = \mathcal{L}_{phoD} + \mathcal{L}_{mrf} + \mathcal{L}_{sym} + \mathcal{L}_{regD} \tag{4.5}$$

$\mathcal{L}_{phoD}$ defines the **photometric detail loss** which, equivalent to the coarse model, aims to reduce the 2D image reconstruction in the face mask V_I. The **Implicit Diversified Markov Random Field (ID-MRF) [36] loss** regularizes the generated content to the original input at a local patch level and thus increases the capabilities to capture high-frequency details. The **Soft symmetry loss** $\mathcal{L}_{sym}$ aims to provide stability when challenged by self-occlusions by regularizing non-visible face parts. The **detail regularization** $\mathcal{L}_{regD}$ regularizes the detail displacements D to reduce noise.

The last step of the tracking pipeline aims to disentangle the dynamic details (represented in the FLAME parameters ψ and θ) from the static details (β). To this end, the authors

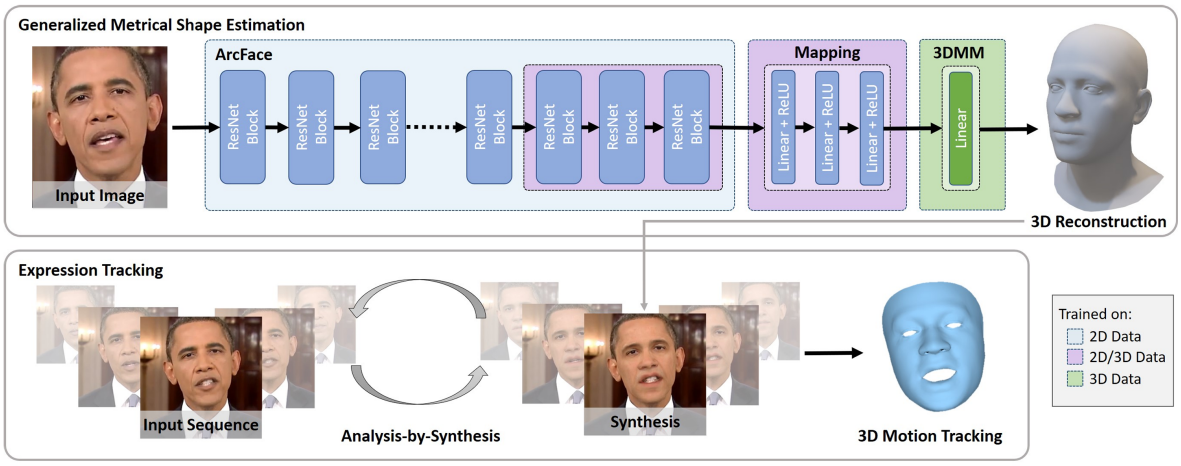


Figure 4.7: The proposed method for metrical human face shape estimation from a single image exploits a supervised training scheme based on a mixture of different 2D, 2D / 3D, and 3D datasets. This estimation can be used for facial expression tracking using Analysis-by-Synthesis, which optimizes for the camera intrinsics, as well as the per-frame illumination, facial expression, and pose.

defined a detail consistency loss between two images I_i and I_j of the same subject. This loss follows the same idea as the shape consistency loss employed in the coarse step but with the difference of swapping the attained detail code δ_i with δ_j

4.2.2 MICA & Metrical Tracker

The MICA (MetrIC fAce) [12] project has the same overall goal as the DECA project: To generate 3D head avatars in FLAME representation from monocular images. However, as the successor project, they obtained even better results than DECA.

They achieved this by building upon the good performance of a state-of-the-art face recognition network (Arcface) [37] but only retraining the last few layers to prevent overfitting. The output of this network is then fed to a geometric Decoder for which the authors employed the FLAME model, resulting in the targeted 3D head avatar. Besides the MICA project, the authors also provide the metrical-tracker extension project, which supports the 3D reconstruction of monocular videos.

MICA - Tracking Pipeline

When tasked with a single monocular image, the MICA project utilizes the power of the pre-trained face recognition network Arcface [37] where the authors fixed the first layers and only trained the last three ones. The output of the retrained Arcface model is then passed through a mapping that produces the FLAME parameters β, ψ, θ. These parameters are then provided to the FLAME geometric Decoder to obtain the full 3D reconstruction.

For the training of the MICA project, the authors used a supervised learning strategy that minimizes the following loss:

$$\mathcal{L} = \sum_{(I,G)\in D} |K_{mask}(G_{3DMM}(M(\text{Arcface}(I))) - G)| \tag{4.6}$$

With G denoting the ground-truth mesh and G_{3DMM} the reconstruction. K_{mask} defines a region-dependent weighting with a significant focus on the frontal part of the face.

Metrical Tracker - Pipeline

In the metrical tracker project, the authors switched from a supervised learning scheme to a self-supervised one by using the Analysis-by-Synthesis scheme of Thiess et al. [38] to model the non-rigid deformations of a face in motion. To ensure shape consistency throughout the generated 3D motion sequence, the authors used the previously defined MICA model as a one-shot shape identity predictor from a defined keyframe of the input video (mostly the first frame). This shape identity is then fixed for the remainder of the sequence. The self-supervised training scheme optimizes the reconstruction of coloring and landmarks using a differentiable renderer.

The landmarks in the input frames are obtained from Google's mediapipe [39] [40] and Face Alignment [41]. The reconstructed key points are obtained from the 3D mesh and the trained camera intrinsics. To improve the processing speed, the authors decided to utilize a pyramid resolution structure, with which the produced 3D model gradually becomes more detailed as the image resolution increases. Overall, with its metrical tracker addition, the MICA project can provide a high-quality 3D motion sequence of head avatars.

4.3 Text Guided Motion Generation

Generating motions for applications such as animation, robotics, and virtual reality can be approached by multiple different model architectures. But, like all generative models, they aim to increase the diversity in generated results by leveraging the non-deterministic behavior of several architectures or training schemes.

4.3.1 Human Motion Diffusion Model

The Human Motion Diffusion Model (HMDM) [1] uses the diffusion-based training approach to achieve non-deterministic behavior. The model consists of a classifier-free Transformer Encoder, which, in contrast to most other diffusion models, predicts the sample x_{start} instead of the noise ε. Besides the text-to-motion, the authors also extended their model to handle action-to-motion and unconditional generation, making it a very versatile trade-off between diversity and fidelity, as the user wants. Besides the full generation, the authors demonstrate motion completion and editing, increasing the range of possible use cases even further.

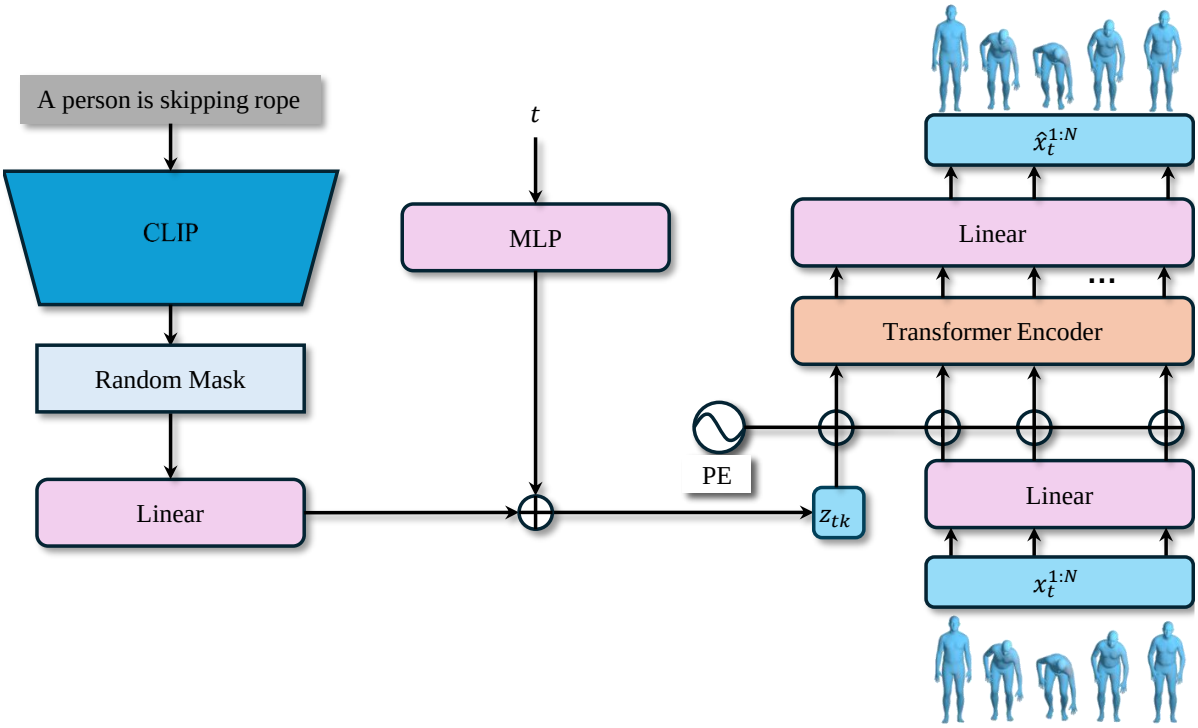


Figure 4.8: **Human Motion Diffusion Model:** The HMDM project encodes the text prompt using the pre-trained Language-to-Image model CLIP and combines it with the encoded parameter t denoting the current diffusion timestep into z_{tk}. z_{tk} is then prepended to the encoded noisy input of the Transformer Encoder. As the resulting sequence is now longer than the desired one, the first element is removed to obtain the de-noised output $\hat{x}_t^{1:N}$

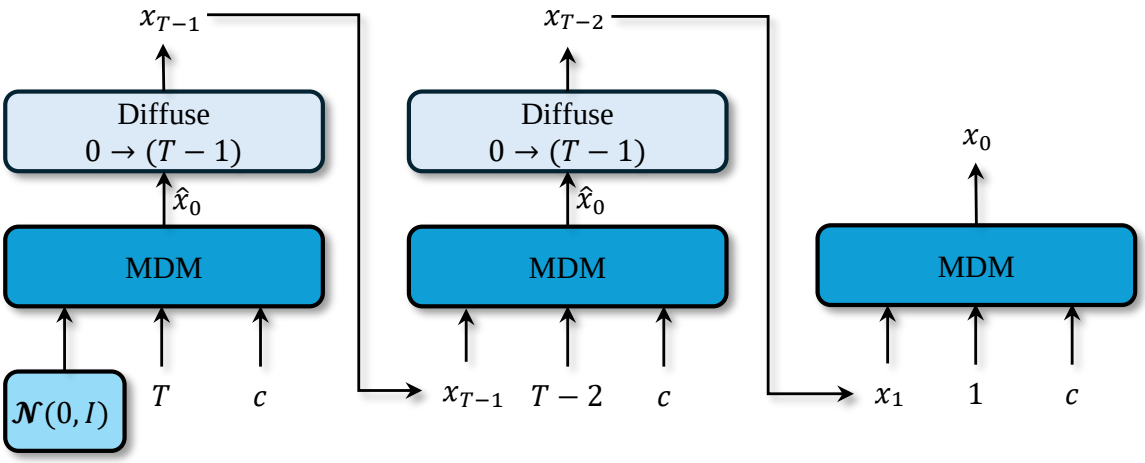


Figure 4.9: Diffusion Process for x_{start} as used in HMDM. Here, the model aims to clean the input fully at every step before the resulting output is again diffused with one less diffusion step. In the model, the default number of denoising steps $T = 1000$ was used, but the authors provided an update to the original work showcasing good results already with $T = 50$ denoising steps.

Figure 4.8 describes the structure of the model with its three inputs: $x_t^{1:N}$ denoting the motion sequence of length N in a noising step t. t denotes the timestep, and c denotes a condition code created from the text input through CLIP [19]. Figure 4.9 describes the sampling process of the model where iteratively from T to 1 the model denoises the input x_t to $\hat{x_0}$ and diffuses it back to x_{t-1} starting with $x_T \sim \mathcal{N}(0, I)$.

The authors used a joint rotation or positional representation to represent the body model correctly. The joint rotation format can be defined as $x^i \in \mathbb{R}^{J \times D}$ with J denoting the number of joints and D the dimension of the joint representation. The positional representation can be described as follows: $x^i \in \mathbb{R}^{3 \times J}$. The authors added support for both representation formats as they utilized different datasets [42] [43] for the experiments. Mainly they used with the rotational representation format.

HMDM is trained using a simple loss (4.7) that calculates the Mean Squared Error (MSE) between the output of the diffusion model at timestep $t-1$ and the ground truth target. In addition to the simple loss, the authors introduced a position (4.8)(in case they predict rotations), foot contact (4.9), and velocity (4.10) loss:

$$\mathcal{L}_{simple} = E_{x_0 \sim q(x_0|c), t \sim [1,T]}[||x_0 - G(x_t, t, c)||_2^2] \tag{4.7}$$

$$\mathcal{L}_{pos} = \frac{1}{N}\sum_{i=1}^{N} ||FK(x_0^i) - FK(\hat{x}_0^i)||_2^2 \tag{4.8}$$

$$\mathcal{L}_{foot} = \frac{1}{N-1}\sum_{i=1}^{N-1} ||(FK(\hat{x}_0^{i+1} - FK(\hat{x}_0^i)) * f_i||_2^2 \tag{4.9}$$

$$\mathcal{L}_{vel} = \frac{1}{N-1}\sum_{i=1}^{N-1} ||(x_0^{i+1} - x_0^i) - (\hat{x}_0^{i+1} - \hat{x}_0^i)||_2^2 \tag{4.10}$$

In the position and foot contact loss, the authors used the Forward Kinematic (FK) function, converting the joint rotations into joint positions. These two losses were added to obtain more accurate motions in their representation format and to prevent the so-called foot-sliding effect. The velocity loss was added to generate more realistic motions with respect to the speed of the specific body parts.

4.3.2 ACTOR

The ACTOR (Action-Conditioned 3D Human Motion Synthesis with Transformer VAE) [3] project aims to tackle the problem of action-conditioned generation of realistic and diverse human full-body motion sequences. In contrast to the HMDM [1] project, it attempts to solve the diversity problem by employing a Transformer Variational AutoEncoder (see 3.3). The proposed Transfomer Variational Autoencoder is split into the Encoder and Decoder block with the stochastical constraint on the latent space.

The Encoder is provided with an arbitrary-length sequence of poses and an action label a as input. It outputs the distribution parameters μ and Σ of the motion latent space. With the reparametrization trick (see 3.3), a latent vector $z \in \boldsymbol{M}$ with $\boldsymbol{M} \subset \mathbb{R}^d$ can be obtained while maintaining the gradient flow to the Encoder. To extract the targeted parameters μ and Σ, the Transformer Encoder's temporal dimension must be pooled. To this end, the authors prepended the inputs of the Encoder with the learnable tokens and only used the corresponding outputs. The two learnable tokens are denoted as μ_a^{token} and Σ_a^{token} and contain the embedded pose sequence.

The Decoder is provided with the latent vector z obtained after the reparametrization trick from μ and Σ, and the action label a. It aims to generate realistic human motion in one shot. The Transformer Decoder block is provided the time information as query (in the form of T sinusoidal positional encodings) and the latent vector combined with the action information as key and value. Here, the authors added a learnable bias b_a^{token} that shifts the latent representation to the action-dependent space. The Decoder block produces a sequence of T vectors in $\mathbb{R}^d$ from which the final poses $\hat{P}_1, ..., \hat{P}_T$ are extracted. Finally, a differentiable

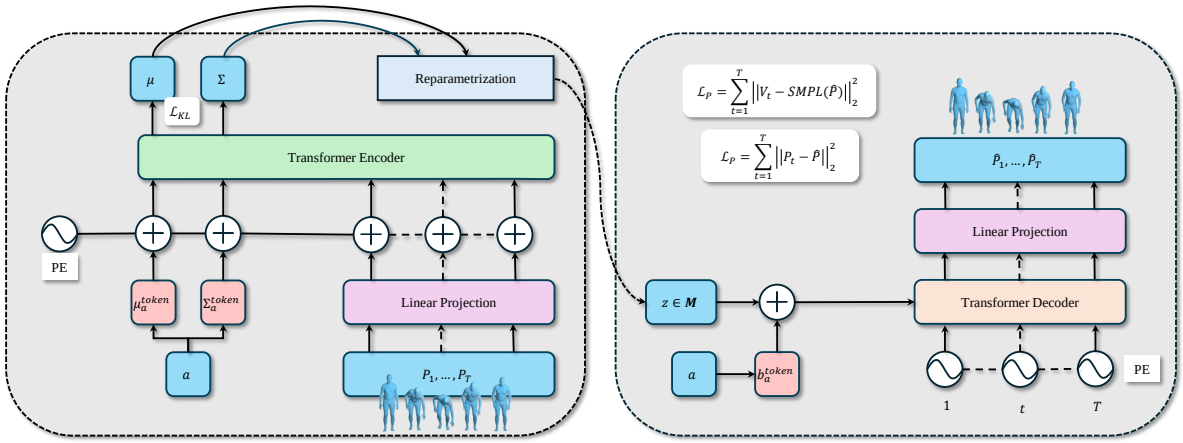


Figure 4.10: **Left:** The transformer-based Encoder is provided a sequence of body poses $P_{1:N}$ together with an action label a. The action label is converted into two learnable input tokens μ_a^{token} and σ_a^{token} that are prepended to the motion sequence before passing through the Transformer Encoder. Afterward, the first two sequential outputs are used as mean μ and standard deviation σ for the reparametrization trick (see 3.3). **Right:** The transformer-based Decoder is provided as input solely the positional encoding of the desired output sequence length. The latent vector z obtained through the reparametrization is combined with another learnable token b_a^{token} extracted again from the action label a and passed to the Transformer Decoder as memory.

SMPL layer converts the poses to a vertex and joint format.

The ACTOR project used three losses to train the model: the Pose Reconstruction Loss, the Vertex Reconstruction Loss, and the KL-Divergence Loss. The **Pose Reconstruction Loss** ($\mathcal{L}_P$) defines the L2 loss between the ground truth poses $P_{1:T}$ and the pose predictions $\hat{P}_{1:T}$. The **Vertex Reconstruction Loss** uses the output after the differentiable SMPL layer and thus calculates the L2 error between the ground truth vertex output $V_{1:T}$ and the generated vertex output $\hat{V}_{1:T}$ The final loss employed to train the ACTOR model is the default **KL-Divergence loss**, ensuring that the trained latent distribution matches the Gaussian Normal distribution as closely as possible. The authors noted that they utilized a weighting λ_{KL} for this loss as it empirically improved their results.

4.4 Guided Facial Motion Generation

Multiple projects exist aimed at generating facial motion with different types of conditioning. Most of them aim at generating lip synchronous motion to an audio input [7] [8] [10] [44]. Especially the FaceFormer [7] project achieved high-quality facial motions, such that we used much of it to inspire our work.
The FaceFormer [7] project attempts to generate lip synchronous facial motion conditioned through an audio input. It follows a Transformer Decoder (see 3.4) and is trained either with an autoregressive or teacher-forcing scheme, but the authors note that their experiments showed better results when training autoregressively.

The left side of Figure 4.11 (Encoder) describes the pre-trained wav2vec [45] Audio-Encoder model. The right side describes the used Transformer Decoder. The Decoder is provided with the past facial motions $\hat{y}_{1:T-1}$, modified by a Periodic Positional Encoding (PPE) (see 3.4), a style embedding s_n uniquely identifying the actor, and the Decoder output $a_{1:kT}$. In addition to the regular inputs, two masks are provided to the Decoder, namely the Temporal Bias $B^{\hat{f}}$ and Alignment Bias B^A.

The Temporal Bias $B^{\hat{f}}$ was introduced to ensure causality and improve the ability to generalize to longer sequences. More specifically, it follows the usual lower triangular structure blocking the model from attending to future sequence elements. Still, it extends upon this idea by weighting the past sequence elements differently, increasing the priority to the more recent sequence elements. It is formulated as:

$$\mathbf{B}^{\hat{\mathbf{f}}}(i,j) = \begin{cases} \lfloor (i-j)/\mathbf{p} \rfloor, \ j \leq i \\ -\infty \ otherwise \end{cases} \tag{4.11}$$

Here the parameter **p** denotes the period hyperparameter also used for the PPE.

The Alignment Bias, B^A, transforms the default Multi-Head attention into a Biased Cross-Modal Multi-Head Attention. It was introduced to combine the Temporal Encoder (speech)

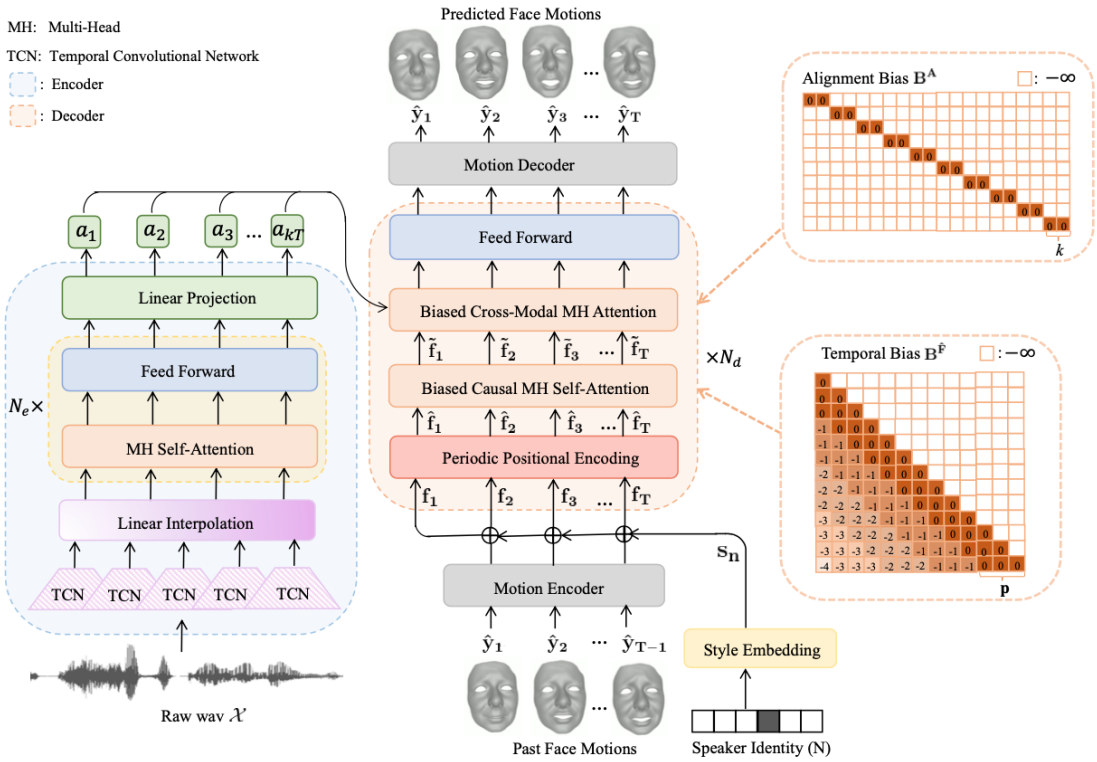


Figure 4.11: **FaceFormer Model:** The **left** side describes the pretrained wav2vec [45] with one additional Linear Projection layer appended. The **right** side describes the proposed Decoder Structure of the FaceFormer project. Instead of the original Sinusoidal Positional Encoding, the authors decided to use Periodic Positional Encoding applied a biased causal mask to the Self-Attention and an alignment bias to the Cross-Attention.

features with the motion features passed through the Multi-Head Self Attention. It is represented as:

$$\mathbf{B}^{\mathbf{A}}(i,j) = \begin{cases} 0 & \text{if } ki \leq j < k(i+1) \\ -\infty & \text{otherwise} \end{cases} \tag{4.12}$$

The model is trained to predict the next frame of the sequence depending on all previously generated frames. This goal can be achieved by training with either a teacher forcing or an autoregressive scheme. The authors of the project supported both variations but noted that in their experiments, the autoregressive format performed better.

The teacher forcing scheme provides the model with the ground truth sequence as input shifted by one position. It is tasked to predict the next sequence element for each element but prevents the shift from being undone by employing a causal mask. A causal mask occludes for

each element of the sequence every subsequent element. One key problem is that the model always gets a "clean slate" as input. Thus, when tasked with generating a new sequence, which has to be done autoregressively, the generated sequence accumulates an error far quicker.

The autoregressive scheme prevents this issue by training and sampling similarly. During training, each sample sequence will be generated iteratively and finally compared against the ground truth. Thus, the model is trained to handle its own inaccuracies. However, this increases the computational complexity and training time manifold as the gradient for the backpropagation algorithm has to be generated iteratively throughout the whole sequence.

The overall optimization goal for the used autoregressive scheme can thus be defined as:

$$\mathcal{L}_{MSE} = \sum_{t=1}^{T} \sum_{v=1}^{V} ||\hat{y}_{t,v} - y_{t,v}||^2 \tag{4.13}$$

With V denoting the number of vertices in the 3D head avatar and T denoting the number of frames in the sequence.

5 Dataset

As seen in Chapter 4, several datasets covering facial motion exist, but none directly match our requirements. Therefore, we decided to generate our own dataset based on CelebV-Text, which provided a large number of natural, in-the-wild facial motions.
In this chapter, we will present our contributions that were made to transform CelebV-Text from a (text, video) dataset to a (text, motion) dataset using the MICA and metrical-tracker projects. We will start by explaining the steps undertaken to select the samples from CelebV-Text, which remained in our newly generated dataset. Afterwards, we will describe the processing steps necessary to transform the dataset from (text, video) pairs to (text, motion) pairs. Finally, we will present the overall pipeline with all steps combined and showcase some results.

5.1 Data Selection Process

Our first step was selecting the samples we wished to use in our dataset. To this end, we analyzed the dataset and defined several rules by which we decided whether or not a sample should be contained. By the nature of the rules, they have to be applied at different steps in the dataset generation pipeline. We will define the rules here but refer to them later when explaining the pipeline step by step.

As seen in 4.1 the CelebV-Text dataset contains a wide variety of videos with heavily varying lengths. With our goal of generating sequences of up to 300 frames at 30 fps in mind, we decided to limit our sequences to lengths between five and ten seconds. With this limitation, we discard about 40% of the samples in the CelebV dataset.

Due to the nature of in-the-wild facial clips, several of the clips contained occlusions of the facial regions, which we did not desire for our dataset. Such occlusions mainly consisted of microphones, facial masks, or sunglasses. We introduced this rule as the generated 3D motion sequence cannot capture occluded facial regions and has to interpolate or guess them, resulting in inaccuracies that could, in turn, worsen the performance of our model. We manually evaluated the videos to ensure this rule is not violated and discarded those we deemed unfit for our dataset.

Similarly to the previous rule, many of the clips were filmed from the side of the person in focus or contained a turning motion of the person. As before, we lose information on the facial motion if the face is not visible, so we decided to discard these samples. We enforced

Figure 5.1: **Discarded Examples of the CelebV-Text dataset:** The **left** clip was discarded as the clip had an overall very low quality, resulting in a high reconstruction error. The **middle** clip was discarded as the mouth was occluded and could thus not be reconstructed. The **right** clip was discarded as it was wrongly cropped, resulting in difficulties when reconstructing

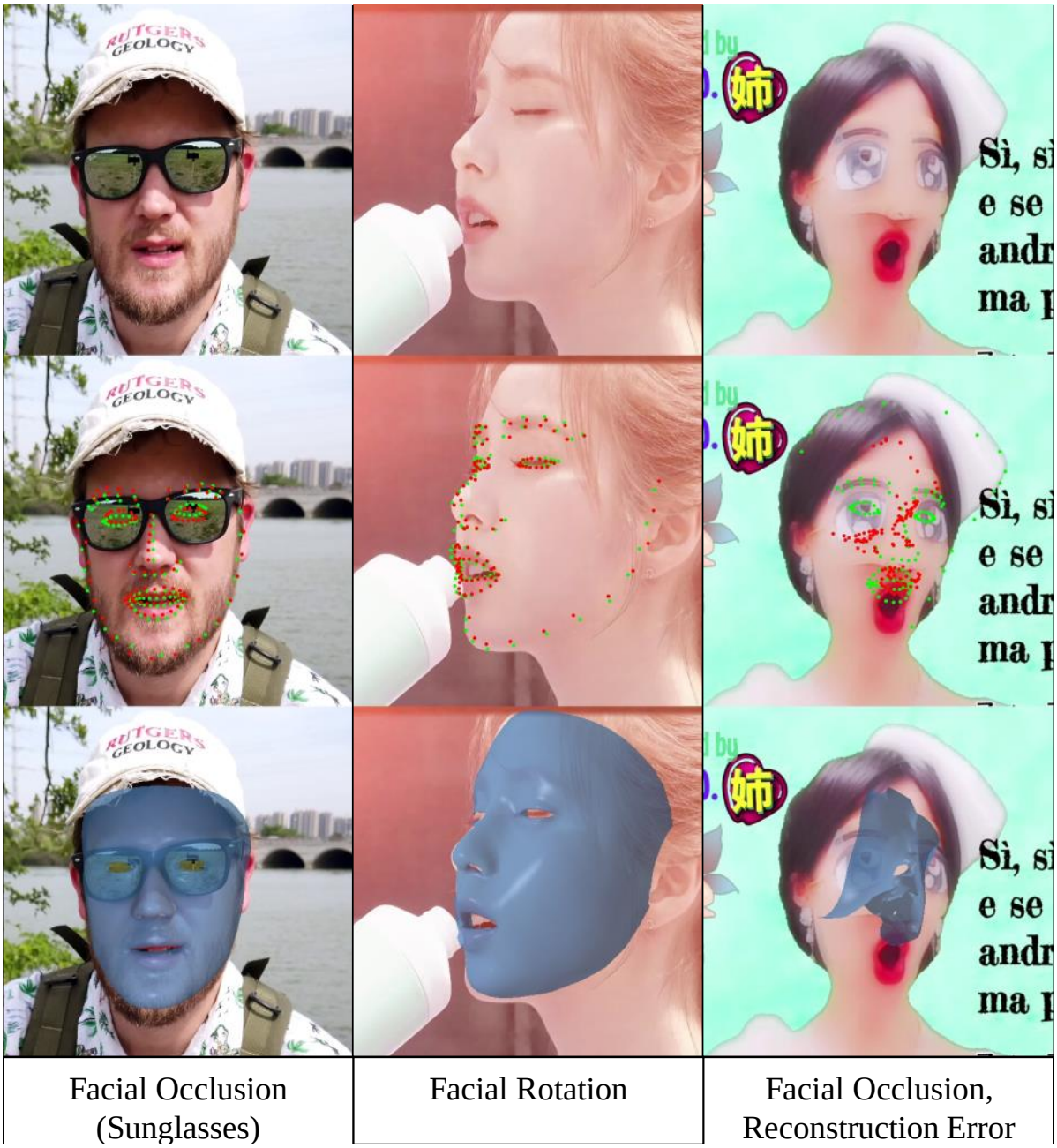


Figure 5.2: **Discarded Examples of the CelebV-Text dataset:** The **left** clip was discarded due to the person wearing reflective sunglasses, resulting in a bad reconstruction for the eye region. The **middle** clip was discarded as the person was turned by a too high degree, rendering the reconstruction of the right side of the face impossible. The **right** clip was discarded as the person was wearing a mask over the eyes, resulting in wrongly predicted facial key points.

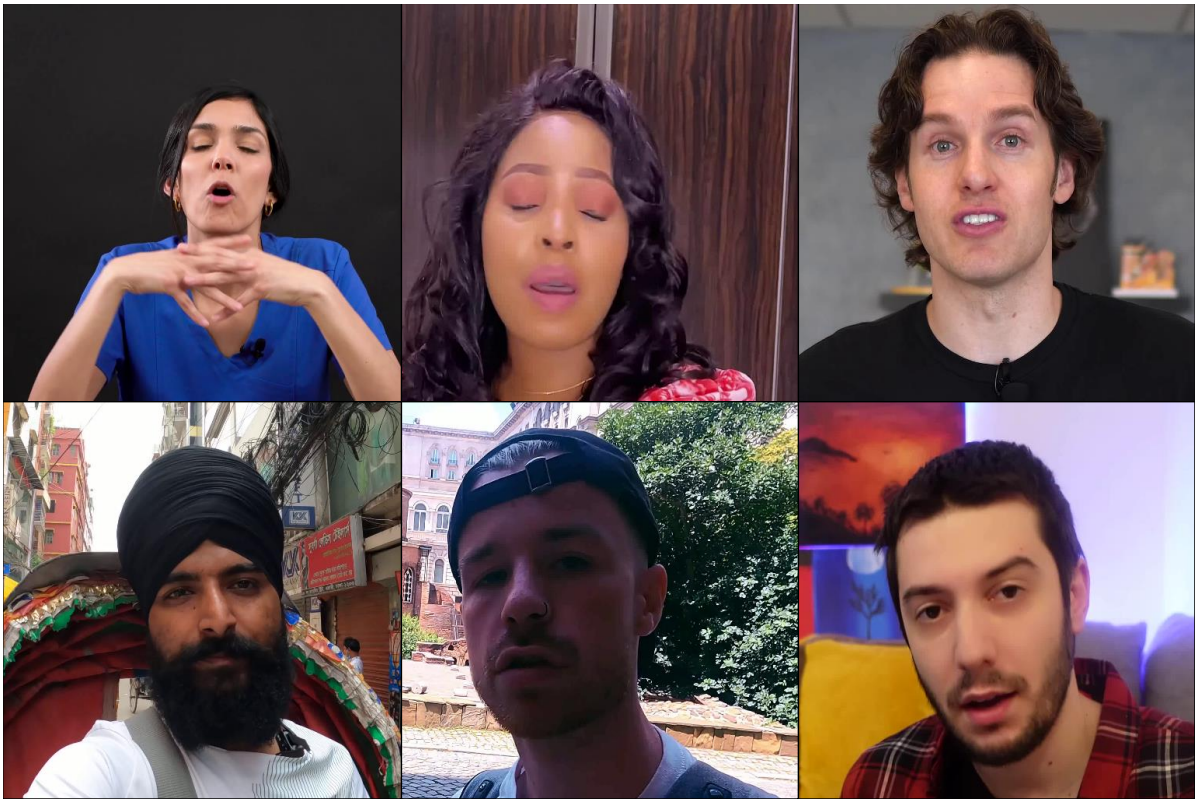


Figure 5.3: **Retained Examples of the CelebV-Text dataset:** This figure showcases some examples retained after applying our criteria to the dataset.

this rule in two steps. Firstly we discarded any sequence that contained the textual annotation "turn", "shake head" and "head wagging". This decision was made as our analysis has shown that most of the sequences labeled with these attributes should be discarded. But as these attributes only describe the dynamic action and not the static position of the camera relative to the person, we employed the 6DRepNet [46] [47] project, which provides a 6D rotation representation for head pose estimation. With the head rotation now known for every frame of the videos, we defined a threshold of 35° in the vertical facial axis. We decided to use 35° as our analysis has shown that we retain samples where parts of the mouth or eye regions are no longer visible for less restrictive boundaries.

The final filtering rule can be considered somewhat of a final failsafe. In contrast to the previous rules, it is applied only during the tracking process and acts as a failsafe if the other rules cannot filter out the unwanted sample. The metrical-tracker project generates the 3D mesh in an Analysis-by-Synthesis manner as described in 4.2. It minimizes the previously introduced loss function between the 3D reconstruction and the 2D input image. To filter out any sequences that could not be reconstructed correctly and thus would harm the quality of the dataset, we applied a loss threshold to the metrical tracker, which a frame must not break for the sequence to be retained. This rule should be able to filter out most of the problematic clips we already catch with the earlier rules, but it does so only after the clip has already been downloaded and potentially partially reconstructed. Therefore, the earlier we filter out unwanted sequences, the fewer resources are lost when discarding a clip.

5.2 Processing Steps

Our processing pipeline is split into four distinct steps: the pre-processing, the video download, the 3D tracking, and the post-processing.

5.2.1 Pre-Processing

In the pre-processing step, we perform our first filtering. Our first filtering consisted of discarding any sequence shorter than five seconds and longer than ten, according to the first rule. In addition to that, we performed rudimentary filtering for too much head movement by using the attribute assignments provided by CelebV. Here we filter out any video containing the actions "turn", "shake head", and "head wagging".

5.2.2 Video Download

With the initial filtering done, we used the provided downloading script of CelebV-Text to download, crop, and cut the videos into the desired clips. As the original script downloads the entire video before performing any crop or cut, we noticed a loss of efficiency as the

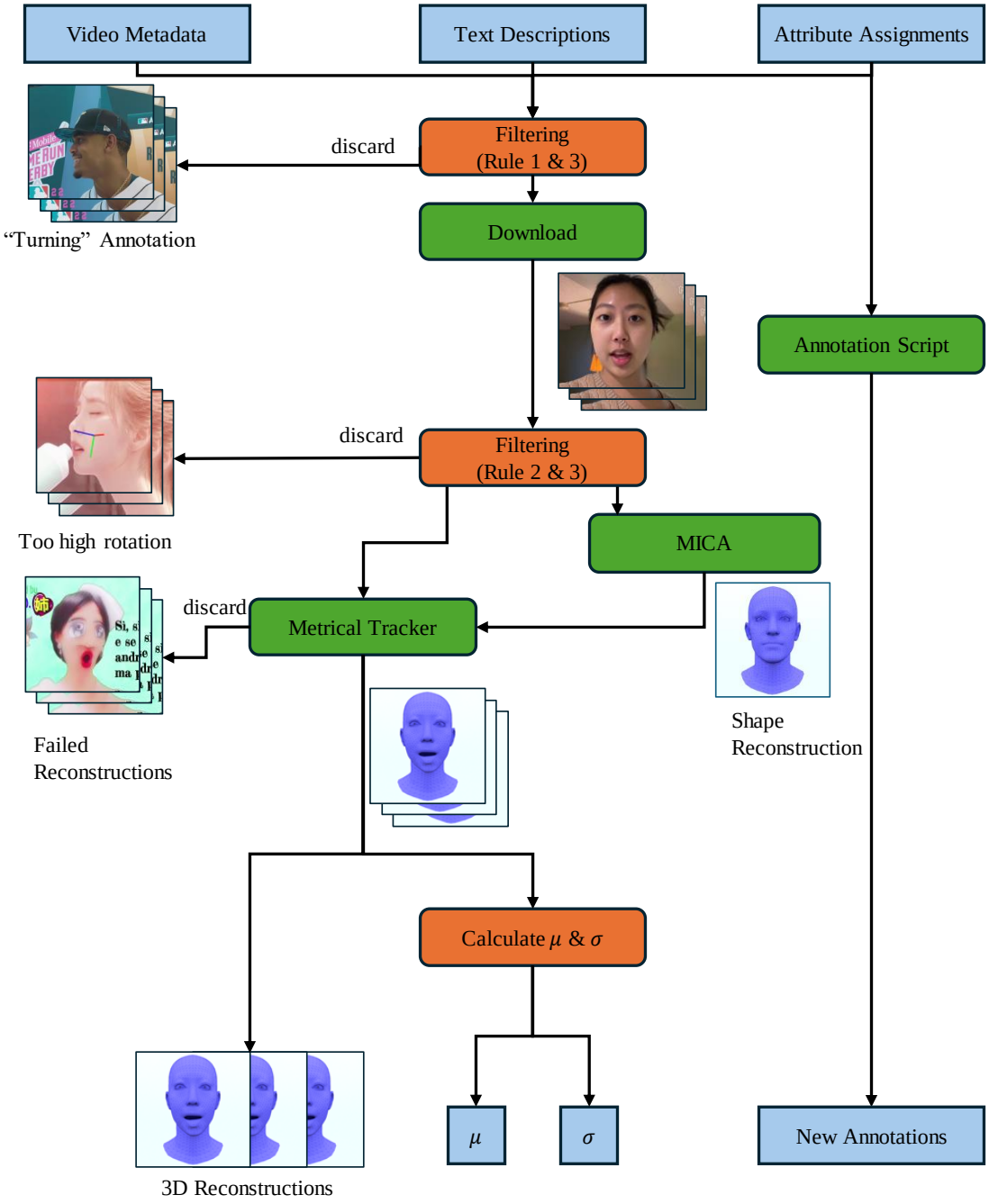


Figure 5.4: **Dataset Pipeline:** This figure illustrates the dataset pipeline. We denote in **blue** the data provided as input or output of our pipeline. In **orange**, we describe the filtering rules as defined in 5.1. The **green** blocks describe the utilized implementations from the CelebV-Text [11], MICA / metrical-tracker [12] projects and our later added annotation script.

downloaded videos often had a length of up to ten minutes, of which we used one (or sometimes multiple) clip(s) of five to ten seconds. We thus experimented with modifying the downloading script to perform the crop on the video stream itself without the necessity of downloading, writing, and then reading everything from the disk. We achieved this goal by replacing the previously used video downloader Aria2c with the video downloader and processing tool FFMPEG [48]. This yielded some increase in efficiency, but only when only one clip was to be obtained from the downloaded video. Here, the download process would be stopped once the end of the targeted clip was reached without requiring intermediate storage on the disk. However, in many cases, the CelebV-Text dataset contained multiple clips in the same original video. In these cases, the original downloading script first downloads the whole video and afterward cuts and crops all of the present clips without requiring more downloading. We thus decided to take the best of both implementations by using our updated downloading script for videos containing only one clip and the original one for videos containing multiple clips.

5.2.3 3D Tracking

After obtaining the clips, we extracted the 3D reconstruction using the MICA project with its metrical-tracker extension. We start by splitting the video clip into different frames using FFMPEG again, ensuring a consistent frame rate of 30 fps. After splitting the videos, we used the 6DRepNet [46] [47] project to obtain the rotation parameters for the x, y, and z axis, respectively. We used these parameters to perform an additional filtering of the dataset using a rotation threshold of 35° in the vertical facial axis.
We then used the first frame of the clip as keyframe input for the MICA project. With this we extracted an actor-unique identity profile describing the shape (in FLAME space) and the keypoints necessary for the tracking of the whole sequence.
In the next step, the Metrical-Tracker model iterates through the frames of the video clip and generates the corresponding 3D reconstruction in the Analysis-by-Synthesis way. They start by scaling down the current image using a Gaussian Pyramid filter and, over multiple training iterations, increase the provided input resolution. After the final training iteration is completed, the 3D reconstruction is saved, and the next frame is processed.
We modified the original project to include our failsafe filter as defined in rule 4 and stopped processing any sequence where the reconstruction could not meet our loss target.

5.2.4 Post-Processing

After we obtained the final 3D reconstruction for each clip, we split the dataset into training, evaluation, and testing parts. For the training data, we used 90% of the data, and for the evaluation and testing, we split 5% each. After that, we calculated the mean and standard deviation of the training data using normalized data in our network.

5.3 Further Modifications

During our work, we encountered a problem with how the authors of CelebV-Text generated the descriptions from the per-frame attribute annotation. The original CelebV-Text description was generated by a probabilistic context-free grammar (PCFG) that was provided a list of attributes and their corresponding start and end timestamps, ordered by the starting timestamps. From this, the PCFG generates the description sequentially. Only if two attributes have the exact same active phase, will they be described as "simultaneous" otherwise the PCFG will describe them as sequential, even if they overlap widely.

This problem can be visualized as seen in Figure 5.5. Here, we demonstrate the mismatch between the provided description and the original binary attribute annotation. As the PCFG sequentially annotates the attributes, ordered by their start time, it does not correctly describe the ground-truth attribute annotation if they co-occur but start or end at different times. To solve this problem, we defined our annotation script. We first prepared the attributes in the script by defining a timeframe whenever an attribute is added or removed. We thus obtain an ordered, sequential list describing exactly which attributes are active in the respective timeframe. We then use this ordered, sequential list in the description generator to obtain the final textual annotation.

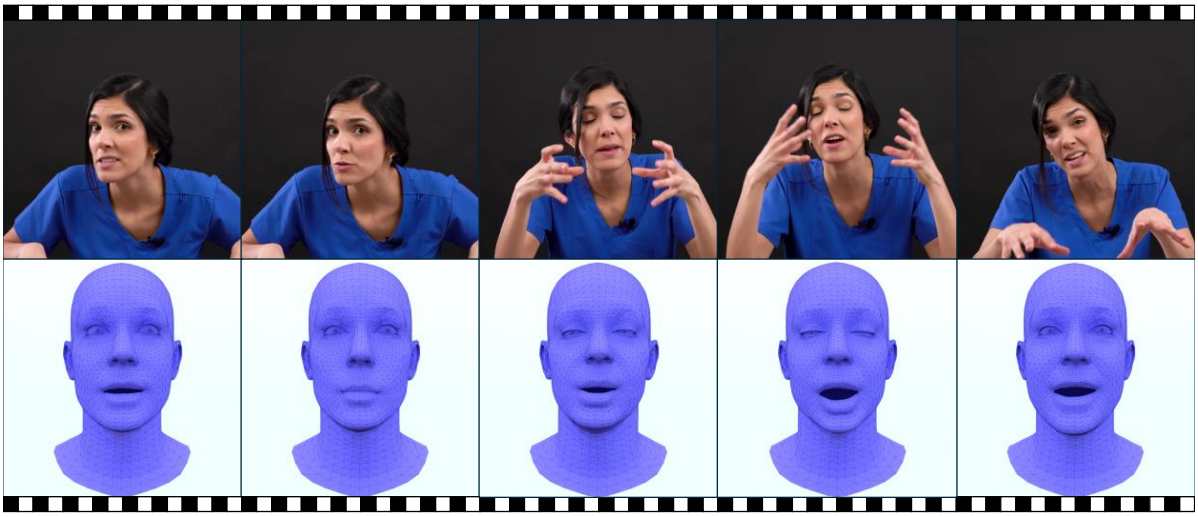
5.4 Efficiency Improving Measures

Due to the largeness of the CelebV-Text dataset, even with our filtering rules applied, we encountered several problems with the required time to process the data. Therefore, we looked into ways to improve the efficiency of our processing pipeline.

The biggest bottleneck in our pipeline was the 3D tracking step, specifically the metrical-tracker project, as it had to perform its tracking steps over every frame of the sequences. To increase the tracking efficiency, we modified the metrical tracker to support multithreading. As the tracker must be run sequentially on the video frames and is self-contained in a Python Class, we achieved this goal by adding a multithreading wrapper, generating multiple instances of the tracker, each processing its separate video.

5.5 Visualizations

Overall, the results of our dataset pipeline can be visualized in Figure 5.6 and 5.7. Here, the top row represents the original input image of the video clip, the middle row is the reconstruction overlayed on the original input, and the bottom row represents the corresponding 3D reconstruction. In the figure, we rendered the 3D reconstructions with the FLAME shape maintained to emphasize the likeness between the 3D reconstruction and the input. In our final dataset, we provide both formats and use only a shape-cleaned version for our proposed method.



Talk, Gaze	Talk, Gaze	Talk, Gaze, Frown	Talk, Gaze, Blink	Talk, Gaze, Blink

Original Description: To begin with, the woman talks meanwhile gazing for a short time, then she frowns for a long time, in the end she blinks for a long time

New Description: To begin with, the woman talks and gazes for a short time, then she talks, gazes and frowns for a short time, in the end she talks, gazes and blinks for a long time

Figure 5.5: **Textual Alignment Problem:** This figure illustrates a discrepancy between the original annotation for video clips in the CelebV-Text dataset and their corresponding per-frame attribute assignments. While the attributes "talk" and "gaze" are persistent throughout the sequence, the original description suggests these attributes are only assigned to the early part of the clip. Our updated annotation script provides a solution to this problem.

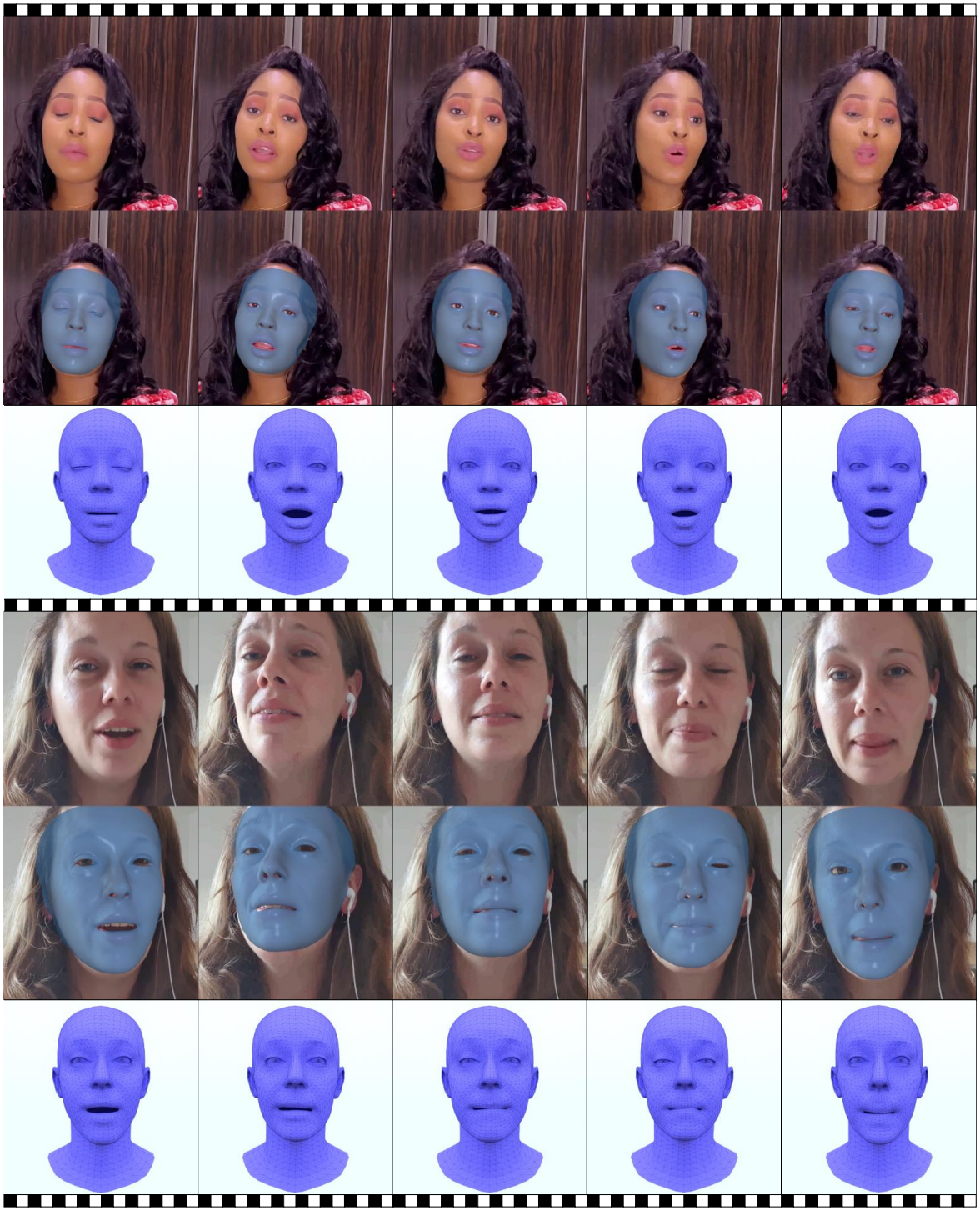


Figure 5.6: **Example Female Clips:** This figure displays several frames of two different clips of female persons, the registered 3D shape rendered atop the original image, and the corresponding unposed 3D reconstruction. To provide a better comparison, the 3D reconstructions were rendered using the shape parameters obtained from the 3D tracking, even if they were removed from the final dataset.

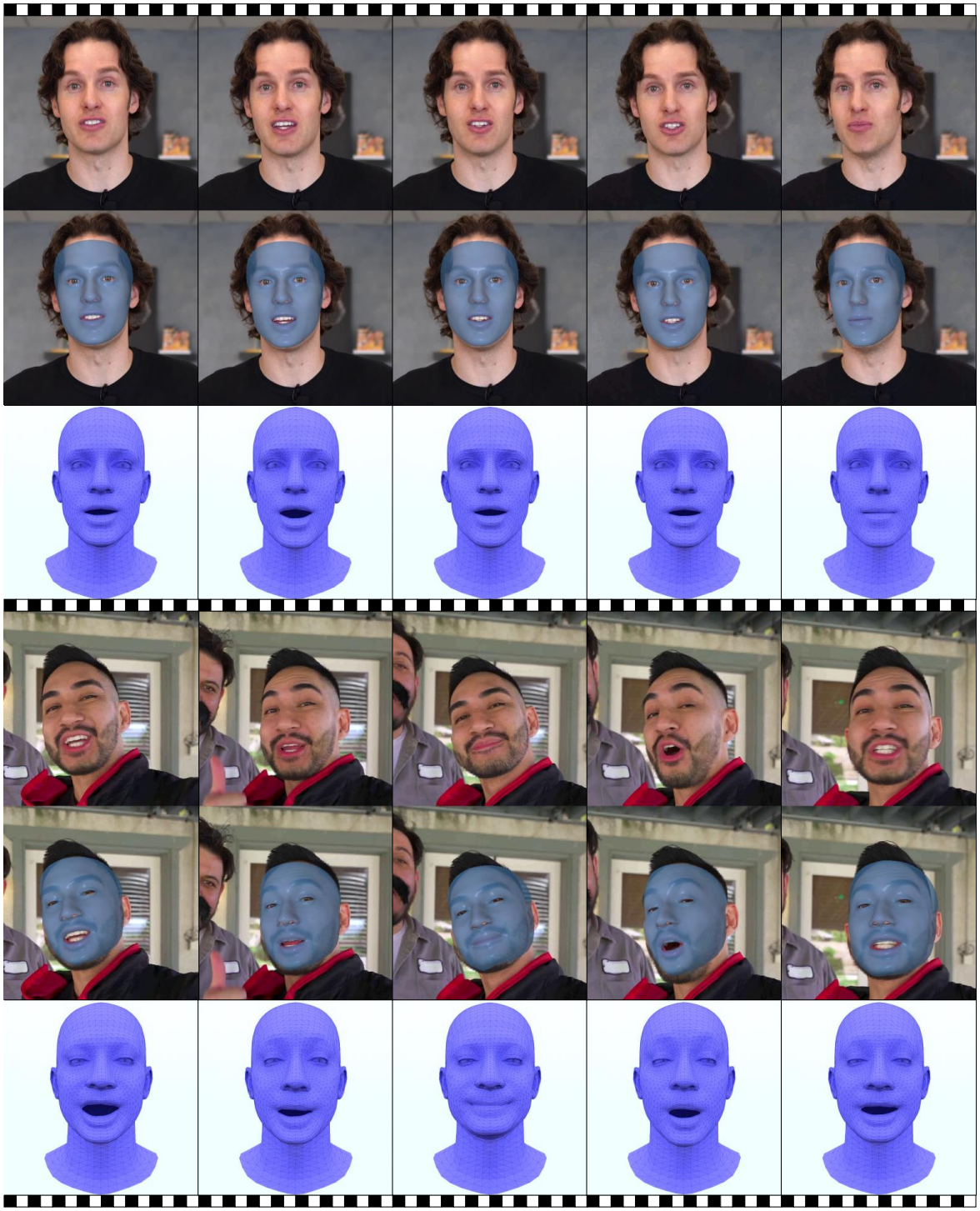


Figure 5.7: **Example Male Clips:** This figure displays several frames of two different clips of male persons, the registered 3D shape rendered atop the original image, and the corresponding unposed 3D reconstruction. To provide a better comparison, the 3D reconstructions were rendered using the shape parameters obtained from the 3D tracking, even if they were removed from the final dataset.

6 Proposed Method

This chapter presents our main work, a model to generate facial motion from a textual action description. We will start by summarizing the existing formats for the input representation and explaining our decision process. Next, we will provide a detailed overview of our proposed model components and cover some possible variations we explored. Finally, we will combine the previously defined components and present the completed model.

6.1 Input Representation

We started our work with the idea of using existing works for text-guided full-body motion generation in the facial domain. Most full-body motion models utilized either a skeletal rotation format or a blend shape format. As a skeletal rotation format by its design provided no compatibility with the facial domain, we started by experimenting with blend shapes, which have been widely used in the facial domain.

In our experiments, our model could not correctly comprehend the nuances obtained through the blend shapes; we experimented on using the FLAME parameters θ and ψ only, as we wished for a shape-independent output) as parameters for our model. Again, our model could not correctly predict the correspondence between the effective motion sequence and the FLAME parameters. We also observed difficulties when using the 6D rotational representation format as introduced by Zhou et al. in [49] for the jaw and eyeball joints.
Thus, we decided to work directly on the vertices of our 3D head avatar as it showed very promising results in [7] [10]. We obtained the shape-cleaned vertices by using the predicted FLAME parameters from our 3D tracking for expression and pose while setting the shape parameters to zero. Our experiments and other works have shown that not working on the absolute vertex position but instead on the vertex offset to a default mesh yielded better results, so we opted to use this approach. Finally, we applied a Z-Normalization to our dataset based on the mean μ and standard deviation σ of the training set for each vertex.
Overall, our model is thus provided with the following parameters:

Motion: Our motion is represented through a $B \times T \times V \times 3$ tensor, with B describing the batch size, T describing the length of the sequence, V the number of vertices. For each vertex, we provide the normalized 3D offset to a base mesh.

Text: Our text condition is represented through a B dimensional array, with every element containing the description corresponding to the element in the batch.

Random Condition: To increase our diversity while maintaining the connection between motion and description, we generated a sample unique random vector of size d_r for each element of the dataset. During the training, this randomness will not be changed over the course of multiple epochs, ensuring consistency between the input and output.

6.2 Model Overview

Our model is split into two main blocks: the Encoder and the Decoder. As Encoder, we employed a conditional Vector Quantized Variational Autoencoder (cVQVAE) in conjunction with the pre-trained Language-to-Image model CLIP to extract diverse guidance throughout the whole sequence. Our Decoder was inspired by the good performance of the FaceFormer [7] project, as it yielded high-quality results when tasked with generating facial motion.

6.2.1 Encoder

Our Encoder block first processes the textual description into a computer-friendly format using the pre-trained Language-to-Image model CLIP (see 3.2) similar to the HMDM project [1]. When employing CLIP as a pre-trained model instead of training any new model from scratch, we can not only increase the efficiency of our model by leveraging the visual information encoded in the latent space of CLIP but also increase our model capability to handle unseen descriptions as CLIP was trained on a far larger language corpus. The CLIP model provides a textual description of at most 77 tokens with an embedding vector of size $d_c = 512$.

Our description embedding vector of shape $B \times d_c$ is then used as the input for a conditional Vector Quantized Variational Autoencoder (cVQVAE). After applying an MLP Encoder, reducing the dimensionality to D_{emb}, we obtained our embedding vector $z_e(x)$. From this embedding vector, the distance to each of the N_{emb} entries in the Codebook is calculated. Using the argmin operator, we obtained the nearest embedding vector. As the argmin operation is non-differentiable, this breaks the gradient flow during the backward pass. To this end, we utilized the proposed straight-through gradient scheme of van den Oord et al., with which the gradient $\nabla_z \mathcal{L}$ obtained from the backward pass to the quantized embedding $z_q(x)$ is passed directly to the original embedding $z_e(x)$, restoring the gradient flow. With this scheme, the gradient can push the output of the MLP Encoder to be discretized differently in the next forward iteration. After obtaining the quantized embedding vector $z_q(x)$, we combined it with our provided conditioning code c, resulting in a vector of size $D_{emb} + d_r$. This vector is then passed through an MLP Decoder, akin to the Encoder, to increase the dimensionality to our desired size. As we desired to extract sequential guidance for our main Decoder block, we increased the dimensionality with our cVQVAE decoder output to a size of $B \times (T \times d_t)$. We then reshaped this output to extract the desired temporal dimension before passing the elements into a Linear Projection Layer, increasing the size of each element from d_t to d_m, our model dimension used in the main Decoder.

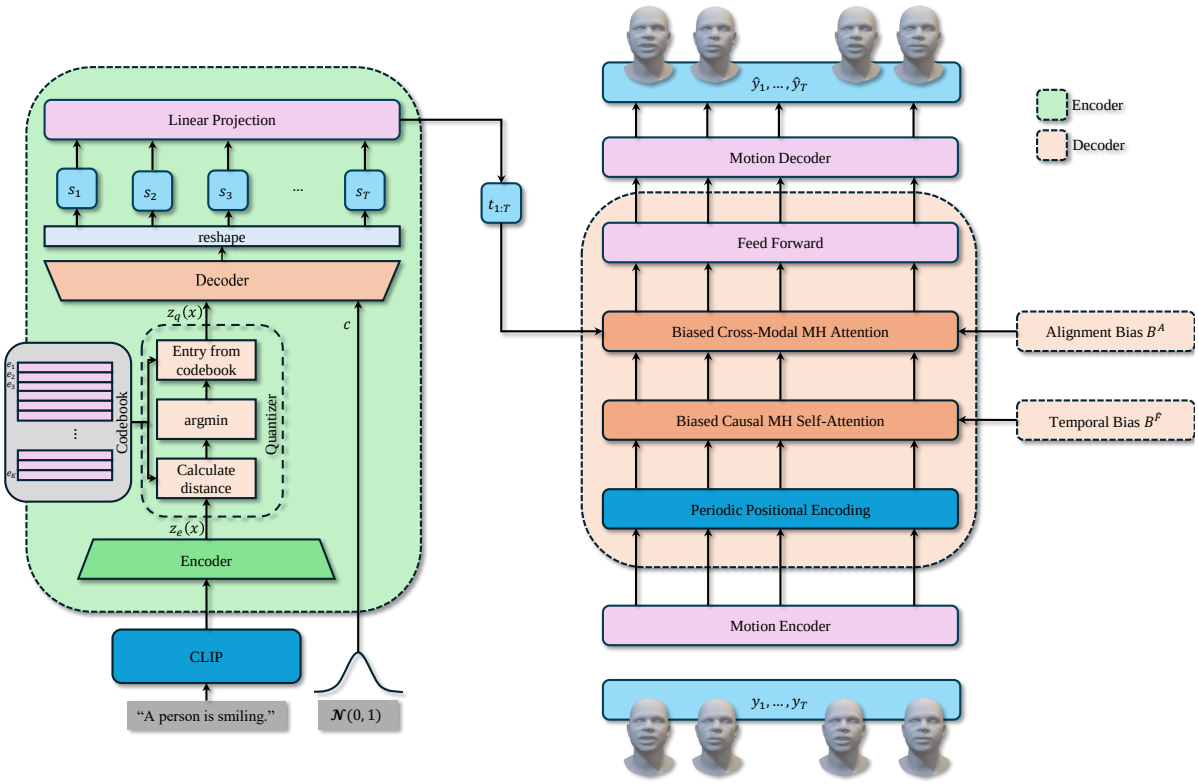


Figure 6.1: **Left:** The raw text will be encoded using the pre-trained CLIP Language-to-Image model, resulting in a d_c dimensional encoding vector. The encoding vector is then fed into an MLP Encoder, producing a D_{emb} sized embedding vector for which the nearest codebook entry is retrieved, combined with the sample's unique randomness, and passed through the MLP decoder. The output of the MLP decoder is then reshaped to obtain a sequential dimension before applying a linear projection to d_m. **Right:** The Transformer Decoder was inspired by the FaceFormer project but decided to train the model with a teacher-forcing scheme instead of an autoregressive one.

6.2.2 Decoder

Our main Decoder block was inspired by the FaceFormer [7] project as they showed promising and high-quality results when tasked with generating facial motions given audio conditioning. However, in contrast to the original work, we decided to train our model not with the autoregressive scheme but with the teacher-forcing scheme, as our experiments showed more promising results for this approach. We refer to Chapter 7, where we compared both approaches as part of an ablation study.

We decided to remove the style conditioning added to every element of the right-shifted input sequence. Instead, we used a zeros vector as the first input element for the Transformer Decoder. We had initially experimented with using and generating a style code using different means but ultimately decided to remove it altogether. We refer to Chapter 7, where we analyzed the impact of utilizing a style conditioning as part of our ablation study.

Our main Decoder input sequence consists of the right-shifted ground truth elements with a zero-vector prepended. The dimensionality of the input is reduced from $B \times T \times V \times 3$ to $B \times T \times d_m$ using a Linear Projection. Afterward, we used the Periodic Positional Encoding with a hyperparameter p denoting the used period. The thus obtained input is passed to a Transformer Decoder with the generated sequential guidance used as Encoder memory, the Temporal Bias $B^{\hat{F}}$ used as the target mask, and the Alignment Bias B^{A} used as memory mask. Finally, we increased the dimensionality from $B \times T \times d_m$ to our input size of $B \times T \times V \times 3$ using a Linear Projection to correctly compute the loss and restore the format for our motion sequence.

6.3 Losses

The optimization goal of our model can be defined as

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{vq} \tag{6.1}$$

With $\mathcal{L}_{vq}$ denoting the **Vector Quantization Loss** and $\mathcal{L}_{rec}$ denoting the **Reconstruction Loss**.

The **Reconstruction Loss** is calculated as the Mean-Squared-Error between the ground truth motion input and the predicted motion output of the Transformer Decoder. As our dataset did not only contain motions of the same length, we used a temporal binary masking m to prevent padding elements from influencing our gradient. We also updated the mean calculation to be consistent with the number of un-padded entries. This results in the following definition for $\mathcal{L}_{rec}$:

$$\mathcal{L}_{rec} = \frac{1}{V \cdot \sum_{i=1}^{T} m_i} \cdot \sum_{i=1}^{T} \left(\sum_{j=1}^{V} (y_{ij} - \hat{y}_{ij}) \cdot m_i \right) \tag{6.2}$$

The **Vector Quantization Loss** forces the Codebook to update the entries to match the Encoder output and updates the Encoder to match the entry of the Codebook better. In this loss, we followed the original implementation by van den Oord et al. [28] with a weighting β and $(1-\beta)$ to the two parts. Overall, the Vector Quantization Loss can be described as

$$\mathcal{L}_{vq} = (1-\beta)||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2 \tag{6.3}$$

With $sg[]$ denoting the stop-gradient operator, restricting the backward gradient flow without hindering the forward pass. By employing the aforementioned straight-through gradient estimation, we enabled our gradient to update the MLP encoder but removed the possibility of the gradient influencing the entries of the Codebook We employed the simple dictionary learning algorithm, Vector Quantization, to alleviate this issue. Here, the Mean-Squared-Error is used to move the embedding entries e_i towards the encoder outputs $z_e(x)$ as seen in the first part of eq. 6.3. As the embedding space itself is dimensionless, it could grow arbitrarily if the embeddings e_i do not train as fast as the MLP Encoder parameters. Thus, the second part of the Vector Quantization Loss is introduced. It acts as a commitment loss, preventing arbitrary growth by ensuring a faster update of the MLP Encoder parameters than the embedding space.

The Reconstruction Loss is applied to all parts of our proposed method except the Codebook. The first part of the Vector Quantization Loss is applied only to the Codebook, and the second part is only applied to the MLP Encoder of the cVQVAE.

7 Results

In this chapter, we will present the results of our proposed method. We will start by defining a baseline evaluation, presenting the results, and then comparing them with other existing works with the same goal.
Afterward, we will compare our baseline results with other results obtained in an ablation study. Here, we will also explain why our baseline architecture performed better than the one used in the ablation study.

7.1 Implementation Details

Our baseline model was trained for 2000 steps of a batch size $B = 8$ using the Adam optimizer on a Nvidia GeForce RTX 2070 GPU with 8 GB of video RAM. Our motion input was thus of shape $8 \times 90 \times 5023 \times 3$ as we used the 5023 vertices of the 3D tracking output. We used the pre-trained CLIP Language-to-Image model "ViT-B/32" using the Vision Transformer architecture described in 3.2. The CLIP model produced an output of size $d_c = 512$, which we used as input for our cVQVAE.

The cVQVAE reduced the input to an embedding size of $D_{emb} = 256$ with $N_{emb} = 128$ entries in the Codebook. Our condition for the cVQVAE was a sample unique random vector of size $d_r = 64$. The Decoder of the cVQVAE produced as output a tensor of shape $B \times (T \times d_t)$ with $d_t = 64$, which we reshaped to obtain the temporal dimension before increasing its size to the latent dimension d_m of the Transformer Decoder.

Our Transformer Decoder used a latent dimension of $d_m = 512$ together with the hyperparameter $p = 15$ denoting the period of the PPE and the Temporal Bias $B^{\hat{F}}$. We decided to use $p = 15$ instead of the original $p = 30$ as described in the FaceFormer project, as our dataset was sampled at only 30 fps in comparison to the 60 fps used in FaceFormer.

While we trained our model using the teacher-forcing scheme, this procedure cannot be used to sample the model. To this end, we swapped to the autoregressive scheme for the generation of new samples. Due to the nature of the autoregressive scheme, we applied a batch size of $B = 1$ during the sampling.

7.2 Baseline Experiments

We will split our baseline study into two parts. The first one will demonstrate the visual output of our model given example prompts. The second one will evaluate and compare the

model to existing works in the same field.

7.2.1 Qualitative Analysis

For our visual study, we performed two experiments. First, we analyzed how well our generated output matched the text description at defined timestamps. Secondly, we analyzed the performance of our model in maintaining a coherent and realistic sequence.

To visually analyze the correlation between our generated model output and the input description, we analyzed the output of the model and its correspondence to the text prompt, abbreviated by the contained attribute, for simplicity reasons. Figure 7.1 demonstrates that our model is capable of producing natural-looking expressions for the different prompts, showing not only a good match to the prompt but also a highly diverse output for similar attributes like "weeping" and "crying," depicting the nuanced understanding of natural language in our model. But this experiment also demonstrates that for prompts with a wide range of possible corresponding motions (i.e., "Talking"), the resulting expression cannot be clearly identified to the attribute as our model was unable to distinguish the different nuances of these actions enough.

To visually analyze the consistency of our model output throughout a whole sequence, we retrieved the first, 20^{th}, 40^{th} and 60^{th} frame of a motion sequence and visualized them together with the text prompt in Figure 7.2. As before, we abbreviated the text prompt to the attribute for simplicity reasons. We decided to depict a sequence with only four frames due to our desire to maintain a high resolution of the images. This experiment demonstrates three main things. Firstly, we observe the first frame of the motion representing the head in a neutral position as our dataset contains motions starting with a neutral position, and our model uses a zero vector as the first sequence element of the Transformer Decoder. Secondly, we observe that the output of our model, after it achieves the desired expression, remains in the expression with only small variations and motions (we refer to our reference implementation for better visualization). Finally, we can observe that the sequential output of our model for a prompt with a wide range of possible corresponding motions ("Talking") could not be generated correctly and has almost no correlation to the prompt.

To qualitatively evaluate our model in comparison to our adapted implementations of the Human Motion-Diffusion Model (HMDM) [1] and StableMoFusion (SMF) [4], we prompted all three models with the same textual description (again abbreviated to the assigned attribute for simplicity reasons) and compared the results of the three models in Figure 7.3. We demonstrate that the HMDM project fails to produce any motion when tasked to generate a facial motion from a text prompt. At the same time, SMF is capable of mainly generating corresponding expressions to the text prompt but fails to maintain the expression for a longer period in the generated sequence. We refer to our reference implementation for a better visualization and video comparison between the three models.

Figure 7.1: **Textual Correctness Study:** This figure illustrates the output of our model for different text prompts. For simplicity reasons, we abbreviated the text prompts to the attributes.

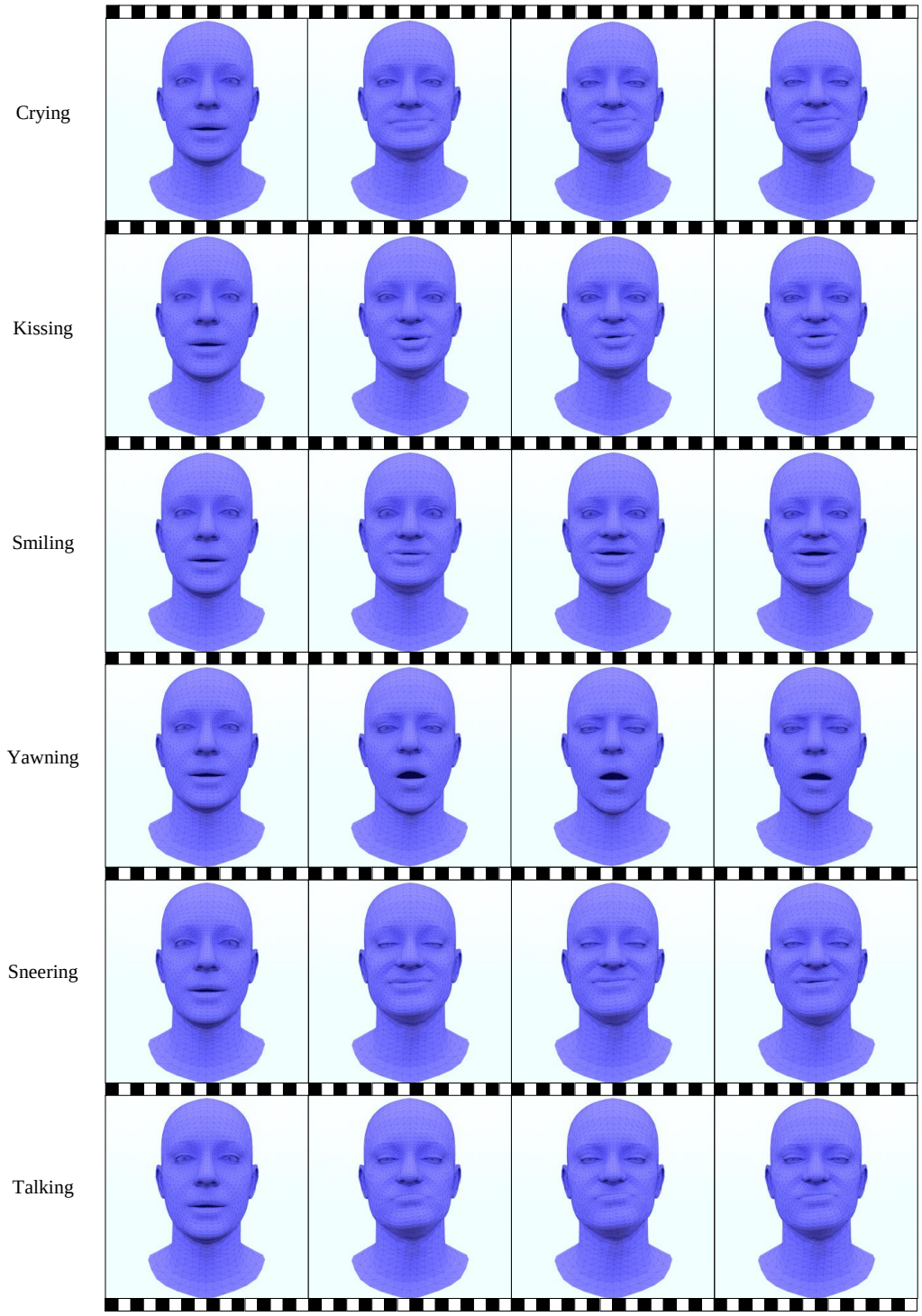


Figure 7.2: **Temporal Consistency Study:** This figure illustrates the model output for different text prompts. We visualize the first, 20^{th}, 40^{th}, and 60^{th} frame of the generated motion sequence. As before, we abbreviated the text prompt to the attribute for simplicity reasons.

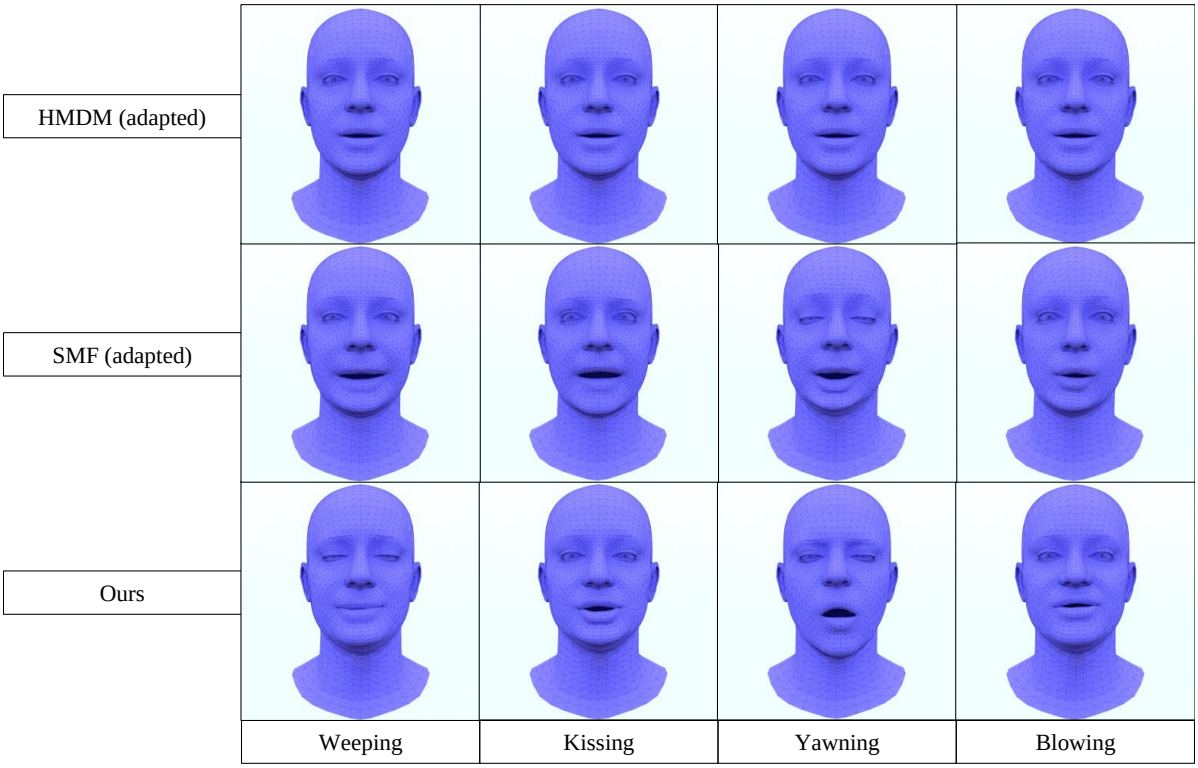


Figure 7.3: **Qualitative Comparison:** We compare the result of our model to samples generated by the adapted versions of the MDM and SMF projects. Here, we show that the MDM project fails to produce any correspondence to the prompt, while SMF is capable of creating good results for some prompts while failing to do so for others. Our model is capable of producing better results even for those where SMF failed.

7.2.2 Quantitative Analysis

As no previous project existed with the same goal as our work, we could not compare it to other works with established metrics. Instead, we decided to compare our project to the public implementation of the Human Motion-Diffusion Model (HMDM) [1] and StableMoFusion (SMF) [4], which we adapted to the facial domain. For the HMDM project, the pre-and post-processing steps were adapted to provide compatibility with our vertex-offset-based input representation while removing the required processing steps for the previously used skeletal input representation. For the SMF project, we added two linear projection layers to reduce and later increase the dimensionality of our input to a more manageable size.

HMDM and SMF evaluated their models using the FID (Frechet Inception Distance), R-Precision, Diversity, and Multimodality. They used a previously existing, pre-trained motion encoder for the FID and Multimodality, as well as a pre-trained contrastive model for the R-Precision. As no such pre-trained models existed for the facial domain and generating such a model from scratch went beyond the scope of this thesis, we instead evaluated and compared our model using the Mean-Absolute-Error (MAE) and Root-Mean-Squared-Error (RMSE) between a generated sample and the ground truth sample from the test split of our dataset. Besides that, we evaluated the Diversity of our model outputs compared to the dataset ground truth and the output of the adapted HMDM and SMF projects. As using the full output of 5023×3 vertices would result in a dilution of expressiveness due to many vertices remaining static (i.e., ears, back of the head, etc.), we decided to select only the vertices of the mouth and eye regions for the quantitative evaluation.

$$E_{MAE} = \frac{1}{N} \sum i = 1^{N} |y_i - \hat{y}_i| \tag{7.1}$$

$$E_{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|^2} \tag{7.2}$$

We assessed all three models over 20 repetitions. In each repetition, we provided the model with the text prompt and assessed the generated motion against the ground truth. Following each repetition, we computed the models' average metrics and evaluated the mean and standard deviation across all repetitions.

As seen in Table 7.1, our model achieves the best performance in terms of MAE and RMSE when comparing our generated output with the ground truth of the dataset. Only in terms of Diversity is it outperformed by the SMF project, but here, it is caused by an inconsistency between generated motion and textual description, as can be seen in the qualitative study (7.3). Especially with respect to the speed of the facial motion does our method produce far more realistic results than the other ones. We refer to our project repository for a sequentially rendered comparison between the models.

Method	MAE ↓	RMSE ↓	Diversity →
Real (Dataset)	-	-	$10.8744^{\pm.0483}$
HMDM (adapted)	$\mathbf{0.6999^{\pm.0000}}$	$\mathbf{0.9006^{\pm.0000}}$	$0.0000^{\pm.0000}$
SMF (adapted)	$0.9272^{\pm.0052}$	$\underline{1.1773}^{\pm.0060}$	$\mathbf{9.2109^{\pm.0620}}$
Ours	$\underline{0.9239}^{\pm.0067}$	$1.2013^{\pm.0083}$	$\underline{7.5996}^{\pm.1755}$

Table 7.1: **Quantitative results on our dataset:** All methods were trained on the same dataset and used the same real motion length from the ground truth data. '→' means the results are better if the metric is closer to the real distribution. We run all evaluation metrics 20 times, and ± indicates the 95% confidence interval. **Bold** indicates the best result. While the HMDM model outperforms our model both in terms of MAE and RMSE, it fails to produce any motion at all but instead generates the average face. The SMF project achieves a higher diversity scoring but often fails to create samples corresponding to the textual description.

Method	MAE ↓	RMSE ↓	Diversity →
Real (Dataset)	–	–	$10.8744^{\pm.0483}$
Autoregressive	$\mathbf{0.7003^{\pm.0000}}$	$\mathbf{0.9008^{\cdot 0000}}$	$0.0431^{\pm.0004}$
Style	$0.9066^{\pm.0047}$	$1.1673^{\pm.0054}$	$6.7660^{\pm.1065}$
Ours (Full)	$0.9239^{\pm.0067}$	$1.2013^{\pm.0083}$	$\mathbf{7.5996^{\pm.1755}}$

Table 7.2: **Ablation Results:** We tested our model with different modifications and ablations. Our reported metrics are evaluated as before.

7.3 Ablation Studies

As defined earlier, our model does not use the style code defined in the original FaceFormer project but instead uses a zero-vector prepended to the shifted ground-truth sequence as input to the Decoder model. Our Style Ablation aims to evaluate this modification by comparing our proposed model with the model using a sample unique random vector as the style code.

Our second experiment aimed to validate the claim of the FaceFormer project that the autoregressive scheme performed better. To this end, we trained our model using the autoregressive scheme with a batch size of 1. As the number of training steps used for the teacher-forcing scheme does not suffice for the autoregressive scheme, we instead trained for a total of 50000 steps.

Table 7.2 compares the first and second ablation experiments to our proposed full model. First, we show that training the model with the autoregressive scheme, as done in the FaceFormer project, resulted in a very bad performance for our model. Second, we show that

D_{emb}	N_{emb}	MAE ↓	RMSE ↓	Diversity →
Real (Dataset)		–	–	$10.8744^{\pm.0483}$
256	256	$0.8872^{\pm.0046}$	$1.1479^{\pm.0051}$	$7.1254^{\pm.1026}$
256	64	$0.9613^{\pm.0052}$	$1.2324^{\pm.0064}$	$6.8904^{\pm.1132}$
128	128	$0.8861^{\pm.0040}$	$1.1376^{\pm.0048}$	$6.8693^{\pm.1042}$
512	128	$\mathbf{0.8830^{\pm.0053}}$	$\mathbf{1.1338^{\pm.0064}}$	$6.4074^{\pm.1032}$
Ours (Full)		$0.9239^{\pm.0067}$	$1.2013^{\pm.0083}$	$\mathbf{7.5996^{\pm.1755}}$

Table 7.3: **Codebook Ablation Study:** We tested our model with different sizes of the cVQVAE Codebook. Our reported metrics are evaluated as before.

using a sample unique randomness vector as style embedding does not provide the aimed increase in Diversity.

Our final Ablation study revolved around the number of VQVAE Codebook entries and their dimensions. We tested both an increase and decrease in the number of Codebook entries and their dimension as they largely influence the quality of guidance upon which our model can be trained. To this end, we trained models where we fixed the dimension of the embeddings to our main model with $D_{emb} = 256$ and increased or decreased the number of embeddings N_{emb}. Our study shows that decreasing the number of embeddings results in an overall worse performance than our proposed method. Increasing the number of embeddings reduces the MAE and RMSE but also reduces the Diversity in generated samples. This better performance in MAE and RMSE is caused by the model being more capable of overfitting on the most average face, as also seen with the HMDM project in our quantitative comparison. Next, we fixated the number of embeddings $N_{emb} = 128$ to that of our proposed model and evaluated the impact of an increase and decrease in the embedding dimension. Again, the increase in the number of embeddings yielded a better performance of MAE and RMSE due to a higher degree of overfitting to the most average face, while a decrease in the number of embeddings resulted in a worse performance in all three measured metrics.

8 Limitations & Future Work

Even if the results of our model trained on our proposed dataset yielded reasonably high-quality results when tasked with text-to-motion generation in the facial domain, some problems and limitations persisted. In this chapter, we will first present some of these limitations we encountered throughout the work. Afterward, we will provide some solutions to them together with other possible improvements that are beyond the scope of this thesis.

Dataset Quality: The CelebV-Text dataset [11] had some problems with the temporal alignment of the text descriptions, as discussed in Chapter 5. With our proposed change to the annotation script, we were able to alleviate this problem somewhat, but two key issues remained:
By the annotation approach of first assigning attributes to the clip and then generating the description from these attributes, the possible space of descriptions is still very limited by the number of attributes used. Especially with attributes like "talk", the range of possible motions was very large, which hampered the performance of our model.
The second key problem consisted of the data distribution in the dataset. The distribution of the attributes throughout the number of samples is far from balanced (see 4.1). With such a large discrepancy between the number of clips for the different attributes, our model was not able to provide good results on some of the attributes as their impact was overwhelmed by other attributes.

Model Diversity: Our model was able to produce reasonable motions for the provided textual description. However, our experiments have also shown that the variability of motions inside one sequence is fairly low, and the variability of sequences with the same description is fairly low. We believe this is caused by the lack of non-determinism in our model compared to the amount of motion achieved in the original HMDM and SMF projects with their diffusion-based approach.

Even if our model can generate naturally-looking motions given text conditioning, the results are far from perfect. To improve upon our work, we propose to explore the following directions:

1. To improve upon our dataset, we propose creating new attribute annotations when sticking to the semi-automatic generation of the annotation or creating new annotations altogether. Here, we propose using the 3D-tracked motion sequence and the corresponding video clip as the basis for the annotation.

2. In this work, we explored different ideas for the generation of facial 3D motion. With our first experiments of simply adapting the HMDM project resulting in failure, we moved away from a diffusion-based approach. Especially when considering our struggle to obtain highly diverse outputs for similar, or even identical, descriptions, another look into diffusion-based models could provide improvements.

3. We proposed using a VQVAE as the basis for our sequence guidance generation as our experiments showed the best results. Petrovich et al. [3] showed that a transformer VAE structure on the whole model could work well with action-to-motion on the full-body domain. It could provide valuable insights to further look into their work and possibly incorporate some of it into our project.

9 Final Remarks

We believe that generative AI models can provide substantial support in the field of low-budget facial motion generation. Here especially, it can open up the stage for many independent or smaller artists, film- and game studios and ease the barrier of entrance for aspiring artists. We also believe it can provide a valuable addition to virtual or augmented artificial intelligence, where the current avatars cannot capture the nuances of facial motions.
To summarize, we can conclude the following points from this work:

1. There have been numerous works aimed at generating full-body motions from textual descriptions [1] [2] [3] [6], but none for the facial domain.
2. This lack of previous projects is most likely caused by the lack of a large and high-quality (text, motion) dataset for head avatars.
3. In this work, we provided a pipeline and dataset that contains a large quantity of high-quality 3D head avatars from in-the-wild video clips.
4. We also provided a model that can generate facial motions from a text prompt, although limited in diversity.

Overall, this project is an early step toward generating realistic and high-quality facial motions from textual guidance. This thesis has shown some promise in the field but needs further exploration.

List of Figures

3.1 **Registration process:** This figure illustrates the necessity to register a 3D surface mesh and its 3D coordinates to a common set of vertices, where each vertex has a specific identity. This can be achieved by a registration process, mapping a canonical mesh template onto each raw 3D face image.[14] 5
3.2 BFM Training Data - Age Distribution 5
3.3 BFM Training Data - Weight Distribution 5
3.4 The mean together with the first three principal components of the shape (left) and texture (right) PCA model. Shown are the mean shape reps. texture plus/minus five standard deviations σ. Masks with the four manually chosen segments (eyes, nose, mouth, and rest) are used in the fitting to extend the flexibility. [15] 6
3.5 Parametrization of the FLAME model (female model shown). **Left**: Activation of the first three shape components between -3 and $+3$ standard deviations. **Middle**: Pose parameters actuating four of the six neck and jaw joints rotationally. **Right**: Activation of the first three expression components between -3 and $+3$ standard deviations.[17] 7
3.6 Joint locations of the female (left) and male (right) FLAME models. Pink/yellow represents right/left eyes, red is the neck joint, and blue is the jaw.[17] 8
3.7 **Summary of the CLIP approach:** While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time, the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes [19] 9
3.8 **VAE Stochastic Goal Formulation [27]:** The VAE learns a stochastic mapping between the observed $\boldsymbol{x}$-space, whose distribution $q_D(\boldsymbol{x})$ is typically complicated, and a latent $\boldsymbol{z}$-space, whose distribution can be relatively simple. The generative model learns a joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with $p_\theta(\boldsymbol{z})$ as prior distribution over the latent space and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ as stochastic decoder. The stochastic encoder $q_\phi(\boldsymbol{z}|x)$ approximates the true, but intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ of the generative model. 11

3.9 **Illustration of the reparameterization trick:** The variational parameters ϕ affect the objective f through the random variable $z \sim q_\phi(z|x)$. We wish to compute gradients $\nabla_\phi f$ to optimize the objective with SGD. In the original form (**left**), we cannot differentiate f w.r.t. ϕ because we cannot directly backpropagate gradients through the random variable z. We can 'externalize' the randomness in z by re-parametrerizing the variable as a deterministic and differentiable function of ϕ, x and a newly introduced random variable ε. This allows us to 'backprob through z', and compute gradients $\nabla_\phi f$. [27] 12
3.10 **Variational Autoencoder:** The Encoder encodes the input into a mean μ and standard deviation σ value. Following the reparametrization trick, these values are used to sample from the predicted distribution and provided to the Decoder. 13
3.11 **Conditional Variational Autoencoder:** In addition to a sampled value from the predicted distribution, a conditioning code c is provided as input. 13
3.12 **Left**: A figure describing the VQVAE. **Right**: Visualization of the embedding space. The Encoder $z(x)$ output is mapped to the nearest point e_2. The gradient $\nabla_z \mathcal{L}$ (in red) will push the Encoder to change its output, which could alter the configuration in the next forward pass. 14
3.13 **Transformer Overview:** This figure illustrates the Transformer architecture as proposed in [29]. **Left** describes one Encoder Layer with its two main blocks, the Multi-Head Self-Attention and a Feed-Forward Network. **Right** describes one Decoder Layer with its three main blocks: a Masked Multi-Head Self-Attention ensuring causality, a Multi-Head Cross-Attention with the memory from the Encoder, and the Feed-Forward Network. All blocks are followed by adding the residual connection and the normalization. Encoder and Decoder Layers are repeated multiple (in [29]: $N = 6$) times. 16
3.14 **Sinusoidal Positional Encoding:** This figure illustrates the Sinusoidal Positional Encoding. Here the x-axis represents the dimension of the model d_{model} and the y-axis the position pos in the sequence. 17
3.15 **Implementation of Rotary Positional Encoding [31]:** This figure illustrates the rotation of the Query and Key matrices by a constant value θ and the value m representing the position of the element in the sequence. 19

4.1 **Overview of CelebV-Text:** CelebV-Text contains (a) 70,000 video samples and (b) 1,400,000 text descriptions. Each video sample is annotated with general appearance, detailed appearance, light conditions, action, emotion, and light direction[11]. 22
4.2 Ethnicity Distribution 23
4.3 Age Distribution 23
4.4 Length Distribution 24
4.5 **FaMoS Example Meshes:** This figure illustrates several poses of the FaMoS dataset from different actors with the goal of depicting a different expression. 26

4.6 **DECA - Tracking Pipeline [35] (Abbreviated):** In DECA, the input image is encoded into parameters for the extraction of an albedo map, a detail map and the FLAME parameters β, θ, ψ which are used to obtain a 3D reconstruction. The 3D reconstruction is then differentiable rendered with obtained camera parameters to obtain a 2D reconstruction of which the reconstruction error to the original image is calculated. 28
4.7 The proposed method for metrical human face shape estimation from a single image exploits a supervised training scheme based on a mixture of different 2D, 2D / 3D, and 3D datasets. This estimation can be used for facial expression tracking using Analysis-by-Synthesis, which optimizes for the camera intrinsics, as well as the per-frame illumination, facial expression, and pose. 30
4.8 **Human Motion Diffusion Model:** The HMDM project encodes the text prompt using the pre-trained Language-to-Image model CLIP and combines it with the encoded parameter t denoting the current diffusion timestep into z_{tk}. z_{tk} is then prepended to the encoded noisy input of the Transformer Encoder. As the resulting sequence is now longer than the desired one, the first element is removed to obtain the de-noised output $\hat{x}_t^{1:N}$ 32
4.9 Diffusion Process for x_{start} as used in HMDM. Here, the model aims to clean the input fully at every step before the resulting output is again diffused with one less diffusion step. In the model, the default number of denoising steps $T = 1000$ was used, but the authors provided an update to the original work showcasing good results already with $T = 50$ denoising steps. 33
4.10 **Left:** The transformer-based Encoder is provided a sequence of body poses $P_{1:N}$ together with an action label a. The action label is converted into two learnable input tokens μ_a^{token} and σ_a^{token} that are prepended to the motion sequence before passing through the Transformer Encoder. Afterward, the first two sequential outputs are used as mean μ and standard deviation σ for the reparametrization trick (see 3.3). **Right:** The transformer-based Decoder is provided as input solely the positional encoding of the desired output sequence length. The latent vector z obtained through the reparametrization is combined with another learnable token b_a^{token} extracted again from the action label a and passed to the Transformer Decoder as memory. 35
4.11 **FaceFormer Model:** The **left** side describes the pretrained wav2vec [45] with one additional Linear Projection layer appended. The **right** side describes the proposed Decoder Structure of the FaceFormer project. Instead of the original Sinusoidal Positional Encoding, the authors decided to use Periodic Positional Encoding applied a biased causal mask to the Self-Attention and an alignment bias to the Cross-Attention. 37

5.1 **Discarded Examples of the CelebV-Text dataset:** The **left** clip was discarded as the clip had an overall very low quality, resulting in a high reconstruction error. The **middle** clip was discarded as the mouth was occluded and could thus not be reconstructed. The **right** clip was discarded as it was wrongly cropped, resulting in difficulties when reconstructing 40
5.2 **Discarded Examples of the CelebV-Text dataset:** The **left** clip was discarded due to the person wearing reflective sunglasses, resulting in a bad reconstruction for the eye region. The **middle** clip was discarded as the person was turned by a too high degree, rendering the reconstruction of the right side of the face impossible. The **right** clip was discarded as the person was wearing a mask over the eyes, resulting in wrongly predicted facial key points. 41
5.3 **Retained Examples of the CelebV-Text dataset:** This figure showcases some examples retained after applying our criteria to the dataset. 42
5.4 **Dataset Pipeline:** This figure illustrates the dataset pipeline. We denote in **blue** the data provided as input or output of our pipeline. In **orange**, we describe the filtering rules as defined in 5.1. The **green** blocks describe the utilized implementations from the CelebV-Text [11], MICA / metrical-tracker [12] projects and our later added annotation script. 44
5.5 **Textual Alignment Problem:** This figure illustrates a discrepancy between the original annotation for video clips in the CelebV-Text dataset and their corresponding per-frame attribute assignments. While the attributes "talk" and "gaze" are persistent throughout the sequence, the original description suggests these attributes are only assigned to the early part of the clip. Our updated annotation script provides a solution to this problem. 47
5.6 **Example Female Clips:** This figure displays several frames of two different clips of female persons, the registered 3D shape rendered atop the original image, and the corresponding unposed 3D reconstruction. To provide a better comparison, the 3D reconstructions were rendered using the shape parameters obtained from the 3D tracking, even if they were removed from the final dataset. 48
5.7 **Example Male Clips:** This figure displays several frames of two different clips of male persons, the registered 3D shape rendered atop the original image, and the corresponding unposed 3D reconstruction. To provide a better comparison, the 3D reconstructions were rendered using the shape parameters obtained from the 3D tracking, even if they were removed from the final dataset. 49

6.1 **Left:** The raw text will be encoded using the pre-trained CLIP Language-to-Image model, resulting in a d_c dimensional encoding vector. The encoding vector is then fed into an MLP Encoder, producing a D_{emb} sized embedding vector for which the nearest codebook entry is retrieved, combined with the sample's unique randomness, and passed through the MLP decoder. The output of the MLP decoder is then reshaped to obtain a sequential dimension before applying a linear projection to d_m. **Right:** The Transformer Decoder was inspired by the FaceFormer project but decided to train the model with a teacher-forcing scheme instead of an autoregressive one. 52

7.1 **Textual Correctness Study:** This figure illustrates the output of our model for different text prompts. For simplicity reasons, we abbreviated the text prompts to the attributes. 57

7.2 **Temporal Consistency Study:** This figure illustrates the model output for different text prompts. We visualize the first, 20^{th}, 40^{th}, and 60^{th} frame of the generated motion sequence. As before, we abbreviated the text prompt to the attribute for simplicity reasons. 58

7.3 **Qualitative Comparison:** We compare the result of our model to samples generated by the adapted versions of the MDM and SMF projects. Here, we show that the MDM project fails to produce any correspondence to the prompt, while SMF is capable of creating good results for some prompts while failing to do so for others. Our model is capable of producing better results even for those where SMF failed. 59

List of Tables

4.1 CelebV-Text 21
4.2 Complete attribute list 23
4.3 Detailed PCFG design for generating descriptions for general faces [11] 24
4.4 Most common action attributes 24

7.1 **Quantitative results on our dataset:** All methods were trained on the same dataset and used the same real motion length from the ground truth data. '$\rightarrow$' means the results are better if the metric is closer to the real distribution. We run all evaluation metrics 20 times, and $\pm$ indicates the 95% confidence interval. **Bold** indicates the best result. While the HMDM model outperforms our model both in terms of MAE and RMSE, it fails to produce any motion at all but instead generates the average face. The SMF project achieves a higher diversity scoring but often fails to create samples corresponding to the textual description. 61
7.2 **Ablation Results:** We tested our model with different modifications and ablations. Our reported metrics are evaluated as before. 61
7.3 **Codebook Ablation Study:** We tested our model with different sizes of the cVQVAE Codebook. Our reported metrics are evaluated as before. 62

Bibliography

[1] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano. "Human Motion Diffusion Model". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=SJ1kSyO2jwu.

[2] G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. *MotionCLIP: Exposing Human Motion Generation to CLIP Space*. 2022. arXiv: 2203.08063 [cs.CV]. URL: https://arxiv.org/abs/2203.08063.

[3] M. Petrovich, M. J. Black, and G. Varol. *Action-Conditioned 3D Human Motion Synthesis with Transformer VAE*. 2021. arXiv: 2104.05670 [cs.CV]. URL: https://arxiv.org/abs/2104.05670.

[4] Y. Huang, H. Yang, C. Luo, Y. Wang, S. Xu, Z. Zhang, M. Zhang, and J. Peng. *StableMoFusion: Towards Robust and Efficient Diffusion-based Motion Generation Framework*. 2024. arXiv: 2405.05691 [cs.CV]. URL: https://arxiv.org/abs/2405.05691.

[5] M. Mori, K. F. MacDorman, and N. Kageki. "The Uncanny Valley [From the Field]". In: *IEEE Robotics & Automation Magazine* 19.2 (2012), pp. 98–100. DOI: 10.1109/MRA.2012.2192811.

[6] C. Ahuja and L.-P. Morency. *Language2Pose: Natural Language Grounded Pose Forecasting*. 2019. arXiv: 1907.01108 [cs.CV]. URL: https://arxiv.org/abs/1907.01108.

[7] Y. Fan, Z. Lin, J. Saito, W. Wang, and T. Komura. "FaceFormer: Speech-Driven 3D Facial Animation with Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[8] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. Black. "Capture, Learning, and Synthesis of 3D Speaking Styles". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10101–10111. URL: http://voca.is.tue.mpg.de/.

[9] T. Bolkart, T. Li, and M. J. Black. "Instant Multi-View Head Capture through Learnable Registration". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 768–779.

[10] S. Aneja, J. Thies, A. Dai, and M. Nießner. *FaceTalk: Audio-Driven Motion Diffusion for Neural Parametric Head Models*. 2024. arXiv: 2312.08459 [cs.CV]. URL: https://arxiv.org/abs/2312.08459.

[11] J. Yu, H. Zhu, L. Jiang, C. C. Loy, W. Cai, and W. Wu. "CelebV-Text: A Large-Scale Facial Text-Video Dataset". In: *CVPR*. 2023.

[12] *Towards Metrical Reconstruction of Human Faces*. 2022.

[13] V. Blanz and T. Vetter. "A Morphable Model For The Synthesis Of 3D Faces". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596730.

[14] P. Koppen, Z.-H. Feng, J. Kittler, M. Awais, W. Christmas, X.-J. Wu, and H.-F. Yin. "Gaussian mixture 3D morphable face model". In: *Pattern Recognition* 74 (2018), pp. 617–628. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0031320317303527.

[15] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. "A 3D Face Model for Pose and Illumination Invariant Face Recognition". In: *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2009, pp. 296–301. DOI: 10.1109/AVSS.2009.58.

[16] B. Amberg, S. Romdhani, and T. Vetter. "Optimal Step Nonrigid ICP Algorithms for Surface Registration". In: June 2007. DOI: 10.1109/CVPR.2007.383165.

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

[18] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. "SMPL: a skinned multi-person linear model". In: *ACM Trans. Graph.* 34.6 (Oct. 2015). ISSN: 0730-0301. DOI: 10.1145/2816795.2818013. URL: https://doi.org/10.1145/2816795.2818013.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[21] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and F.-F. Li. *Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations*. 2016. arXiv: 1602.07332 [cs.CV]. URL: https://arxiv.org/abs/1602.07332.

[22] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. "YFCC100M: the new data in multimedia research". In: *Communications of the ACM* 59.2 (Jan. 2016), pp. 64–73. ISSN: 1557-7317. DOI: 10.1145/2812802. URL: http://dx.doi.org/10.1145/2812802.

[23] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[24] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV]. URL: https://arxiv.org/abs/1911.05722.

[25] R. Zhang. *Making Convolutional Networks Shift-Invariant Again*. 2019. arXiv: 1904.11486 [cs.CV]. URL: https://arxiv.org/abs/1904.11486.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[27] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. ISSN: 1935-8245. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.

[28] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. "Neural Discrete Representation Learning". In: *CoRR* abs/1711.00937 (2017). arXiv: 1711.00937. URL: http://arxiv.org/abs/1711.00937.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[30] O. Press, N. Smith, and M. Lewis. "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=R8sQPpGCv0.

[31] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. "RoFormer: Enhanced Transformer with Rotary Position Embedding". In: *CoRR* abs/2104.09864 (2021). arXiv: 2104.09864. URL: https://arxiv.org/abs/2104.09864.

[32] P. Shaw, J. Uszkoreit, and A. Vaswani. *Self-Attention with Relative Position Representations*. 2018. arXiv: 1803.02155 [cs.CL]. URL: https://arxiv.org/abs/1803.02155.

[33] H. Zhu, W. Wu, W. Zhu, L. Jiang, S. Tang, L. Zhang, Z. Liu, and C. C. Loy. "CelebV-HQ: A Large-Scale Video Facial Attributes Dataset". In: *ECCV*. 2022.

[34] B. Castellano. *PySceneDetect*. URL: https://github.com/Breakthrough/PySceneDetect.

[35] Y. Feng, H. Feng, M. J. Black, and T. Bolkart. "Learning an Animatable Detailed 3D Face Model from In-The-Wild Images". In: vol. 40. 8. 2021. URL: https://doi.org/10.1145/3450626.3459936.

[36] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia. *Image Inpainting via Generative Multi-column Convolutional Neural Networks*. 2018. arXiv: 1810.08771 [cs.CV]. URL: https://arxiv.org/abs/1810.08771.

[37] J. Deng, J. Guo, and S. Zafeiriou. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *CoRR* abs/1801.07698 (2018). arXiv: 1801.07698. URL: http://arxiv.org/abs/1801.07698.

[38] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. *Face2Face: Real-time Face Capture and Reenactment of RGB Videos*. 2020. arXiv: 2007.14808.

[39] I. Grishchenko, A. Ablavatski, Y. Kartynnik, K. Raveendran, and M. Grundmann. "Attention Mesh: High-fidelity Face Mesh Prediction in Real-time". In: *CoRR* abs/2006.10962 (2020). arXiv: 2006.10962. URL: https://arxiv.org/abs/2006.10962.

[40] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs". In: *CoRR* abs/1907.06724 (2019). arXiv: 1907.06724. URL: http://arxiv.org/abs/1907.06724.

[41] A. Bulat and G. Tzimiropoulos. "How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230, 000 3D facial landmarks)". In: *CoRR* abs/1703.07332 (2017). arXiv: 1703.07332. URL: http://arxiv.org/abs/1703.07332.

[42] C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng. "Generating Diverse and Natural 3D Human Motions From Text". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5152–5161.

[43] M. Plappert, C. Mandery, and T. Asfour. "The KIT Motion-Language Dataset". In: *CoRR* abs/1607.03827 (2016). arXiv: 1607.03827. URL: http://arxiv.org/abs/1607.03827.

[44] Y. Ma, S. Wang, Y. Ding, B. Ma, T. Lv, C. Fan, Z. Hu, Z. Deng, and X. Yu. *TalkCLIP: Talking Head Generation with Text-Guided Expressive Speaking Styles*. 2023. arXiv: 2304.00334 [cs.CV]. URL: https://arxiv.org/abs/2304.00334.

[45] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *CoRR* abs/2006.11477 (2020). arXiv: 2006.11477. URL: https://arxiv.org/abs/2006.11477.

[46] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi. "Toward Robust and Unconstrained Full Range of Rotation Head Pose Estimation". In: *IEEE Transactions on Image Processing* 33 (2024), pp. 2377–2387. DOI: 10.1109/TIP.2024.3378180.

[47] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi. "6d Rotation Representation For Unconstrained Head Pose Estimation". In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 2496–2500. DOI: 10.1109/ICIP46576.2022.9897219.

[48] S. Tomar. "Converting video formats with FFmpeg". In: *Linux Journal* 2006.146 (2006), p. 10.

[49] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. *On the Continuity of Rotation Representations in Neural Networks*. 2020. arXiv: 1812.07035.