BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

THE IMPACT OF FACIAL EXPRESSIONS ON 3D FACE RECOGNITION

VLIV VÝRAZŮ OBLIČEJE NA 3D ROZPOZNÁVÁNÍ PODLE OBLIČEJE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **PETER KOVÁČ**
AUTOR PRÁCE

SUPERVISOR **Ing. FILIP PLEŠKO**
VEDOUCÍ PRÁCE

BRNO 2024

Abstract

The thesis "The impact of facial expressions on 3D face recognition" focuses on studying the effect of facial expressions on the accuracy of 3D face recognition. The first chapters explore various 3D modeling techniques and their applications, including 3D Morphable Models (3DMMs), blendshape models, and neural network-based methods like FLAME (Faces Learned with an Articulated Model and Expressions). The thesis then presents a new approach for reconstructing 3D face models from 2D images and proposes an evaluation framework to measure the impact of facial expression changes on 3D face recognition. Experimental results demonstrate how different expressions, such as anger, happiness, and fear, influence recognition accuracy. The findings of this work contribute to a deeper understanding of facial expressions' role in 3D face recognition and propose potential improvements for enhancing recognition systems.

Abstrakt

Bakalárska práca "Vplyv výrazov tváre na rozpoznávanie tváre v 3D" sa zameriava na štúdium vplyvu výrazov tváre na presnosť rozpoznávania tváre v 3D. Prvé kapitoly skúmajú rôzne techniky 3D modelovania a ich využitie vrátane 3D morfovateľných modelov (3DMM), modelov blendshape a metód založených na neurónových sieťach ako FLAME (Faces Learned with an Articulated Model and Expressions). Práca následne predstavuje nový prístup k rekonštrukcii 3D modelov tvárí z 2D obrázkov a navrhuje hodnotiaci rámec na meranie vplyvu zmien výrazov tváre na rozpoznávanie tváre v 3D. Experimentálne výsledky ukazujú, ako rôzne výrazy, ako napríklad hnev, radosť a strach, ovplyvňujú presnosť rozpoznávania. Zistenia tejto práce prispievajú k hlbšiemu pochopeniu úlohy výrazov tváre v rozpoznávaní tváre v 3D a navrhujú možné zlepšenia na zvýšenie presnosti rozpoznávacích systémov.

Keywords

3D face recognition, facial expressions, 3D Morphable Models, blendshape models, FLAME, DECA, landmark detection, 3D modeling, facial animation

Klíčová slova

3D rozpoznávanie tvárí, výrazy tváre, 3D morfovateľné modely, modely blendshape, FLAME, DECA, detekcia bodov, 3D modelovanie, animácia tváre

Reference

KOVÁČ, Peter. *The impact of facial expressions on 3D face recognition.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Filip Pleško

Rozšířený abstrakt

Cieľom tejto bakalárskej práce bolo zamerať sa na vplyv výrazov tváre na 3D rozpoznávanie podľa tváre. Touto problematikou sa zaoberá kapitola 5. V úvodných sekciách sú popísané rôzne metódy na zber 3D dát, existujúce metódy na vytváranie 3D modelov, z ktorých najzaujímavejšie sú popísané do väčších detailov.

V kapitole 3 bolo cieľom navrhnúť riešenie na 3D rekonštrukciu tváre, teda vytvorenie 3D modelu tváre z obrázku. Ako sa ukázalo v implementačnej fáze, riešenie nie je dostačujúce na reálne výsledky. Návrh pozostáva z týchto častí: získanie datasetu ľudských tvárí, nájdenie metódy na získanie bodov na ľudskej tvári, neurónová sieť, ktorá vie z obrázka na vstupe predpovedať parametre do FLAME modelu, FLAME dekodér pre vytvorenie 3D modelu na základe predpovedaných parametrov, projekcia vygenerovaného modelu do 2D a porovnanie bodov vstupného a výstupného obrázka.
Pri vytváraní návrhu sa riešenie javilo ako uskutočniteľné, ale počas implementácie sa zistilo, že model má nedostatok dát na presnú predikciu FLAME parametrov. Napriek tomu, že graf loss funkcie ukazuje klesajúci trend, výsledný model tváre vykazuje veľmi nízku podobnosť so vstupným obrázkom. Príčinou môže byť to, že počas tréningu neurónovej siete sa FLAME parametre nezlepšovali a model sa namiesto toho naučil "oklamať" algoritmus na detekciu bodov na tvári, aby sa zhodovali s bodmi na vstupnej tvári.

Jedným z problémov pri takto komplexnom riešení je zabezpečiť vzájomnú kompatibilitu jednotlivých komponentov. Je dôležité, aby boli všetky časti od vstupu do neurónovej siete až po detekciu bodov na výslednej tvári diferencovateľné, čo znamená, že z nich musíme byť schopní počítať gradienty. Pri hľadaní riešenia na diferencovateľnú 3D projekciu bolo vyskúšaných päť rôznych repozitárov, z ktorých nakoniec fungoval iba jeden (nekompatibilita knižnice TensorFlow s ostatnými knižnicami, prípadne s verziou CUDA, alebo nemožnosť inštalácie). Prínos výsledkov v kapitole 4 je prakticky nulový, avšak sú popísané možné dôvody neúspechu a teoretické vylepšenia do budúcnosti.

V kapitole 5 sa podarilo predložiť reálne výsledky. Jej cieľom bolo vygenerovať niekoľko výrazov tváre a následne porovnať ich vplyv na 3D rozpoznávanie podľa tváre. Boli otestované tri experimenty. Prvý experiment sa zameral na presnosť 3D rozpoznávania tváre pomocou neutrálnych výrazov. Model sa naučil presne kategorizovať tváre už za 50 epôch pre všetky kategórie. Druhý experiment skúmal, či model dokáže rozpoznať tváre aj napriek zmenám vo výraze. Výsledky ukázali, že model nedosahuje vysokú presnosť pri identifikácii tvárí so zmenenými výrazmi. Zvýšenie počtu bodov mierne zlepšilo presnosť, no aj pri 8192 bodoch sa presnosť zastavila na pomerne nízkej úrovni. Tretí experiment zahŕňal tréning modelu na kombinácii neutrálnych a zmenených výrazov, aby sa zistilo, či použitie viacerých variácií zvýši presnosť. Ako sa očakávalo, výsledky boli sľubné a model dosiahol výrazne vyššiu presnosť. Napriek tomu, že tréning trval dlhšie, zahrnutie viacerých variácií výrazov zvýšilo presnosť významne. Avšak model dosiahol plateau pri približne 60% presnosti, čo naznačuje, že presnosť by sa už nezvyšovala pridaním ďalších bodov.

Hypotéza, že poskytnutie väčšej diverzity pre systémy 3D rozpoznávania zvýši ich presnosť, bola potvrdená.

The impact of facial expressions on 3D face recognition

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Filipa Pleška... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Peter Kováč

May 9, 2024

Contents

List of Figures

Chapter 1

Introduction

The idea of 3D face reconstruction from visual input has been around for several decades. This process involves taking a flat (2D) image of a human face as input and turning it into a 3D representation. Computer software looks at this picture and tries to figure out how the face would look from different angles. This includes guessing the shape of the nose, depth of the eyes, etc. The result is a 3D model which can be rotated and viewed from different angles. In the early chapters, I'll explore the topic of 3D modelling itself, changing and animating facial expressions using different techniques. Later in the thesis, the focus will be on measuring the effects of changing facial expressions in 3D face recognition. I believe the prerequisite for this is a deep understanding of three-dimensional modelling of faces.
In Chapter 2, I'll examine previous papers aiming to model and reconstruct facial structures. The main focus will be on 3DMMs, specifically the paper by Vetter and Blanz [2], which laid a foundation in the field of 3D face reconstruction and changing facial expressions. The section on 3DMM also includes the used dataset, the parametrization of face space, Principal Component Analysis (PCA) and expression modification. Other techniques, such as blend shapes, physics-based methods and the use of neural networks in this field, are also mentioned. In Chapter 3, I propose my solution, inspired by existing solutions, to reconstruct a 3D model from a single image. Chapter 4 comprises implementing the proposed approach. The aim of the last part of the thesis, chapter 5, is to study the impact of changing facial expressions in the context of 3D face recognition. Further in this section, I'll briefly mention the most common 3D data acquisition techniques.

1.1 3D data acquisition techniques

This section contains various techniques used in data acquisition for 3D face modelling. I included this part in the first chapter, as obtaining training data is an elementary prerequisite for any 3D reconstruction method. The following paragraphs are inspired by *„3D Morphable Face Models—Past, Present, and Future“* [3].

There are various methods used for creating training data for 3D reconstruction techniques.

Geometric methods

We distinguish between two types of geometric methods - active and passive. **Active** methods emit light or other signals onto the scanned surface. The light that interacts with the object is then reflected and captured by a sensor.

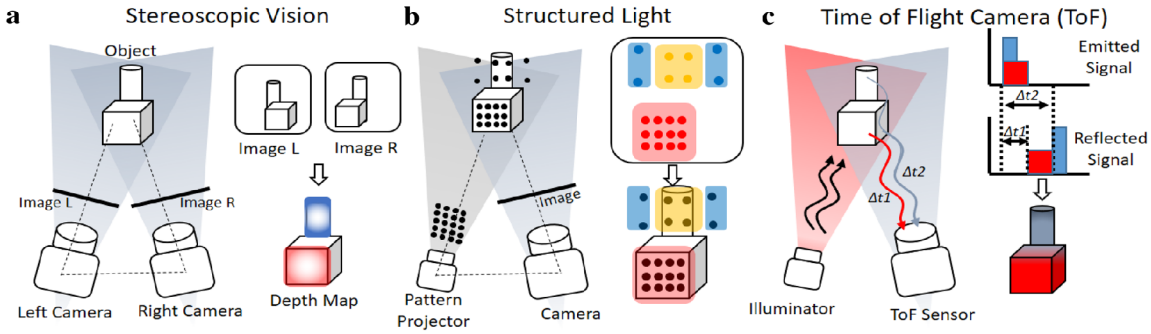


Figure 1.1: Visual representation of various 3D mapping techniques. Adapted from *Optical MEMS devices for compact 3D surface imaging cameras (Yang et al., 2019)* [24]

The concept of **Time-of-flight** (ToF) cameras has been around for quite some time, but has only recently gained popularity due to technological advancements. ToF cameras are used in various fields, such as robot navigation, vehicle monitoring, people counting, object detection, and 3D modelling. The key idea is using infrared light outside the visible spectrum, making the scanning less susceptible to ambient light. The process involves firing a burst of infrared light (photons) towards the target. Pulses which are not absorbed are reflected and captured by the ToF sensor. To calculate the distance to the target, we can use a simple formula:

$$d = \frac{v_l t}{2} \tag{1.1}$$

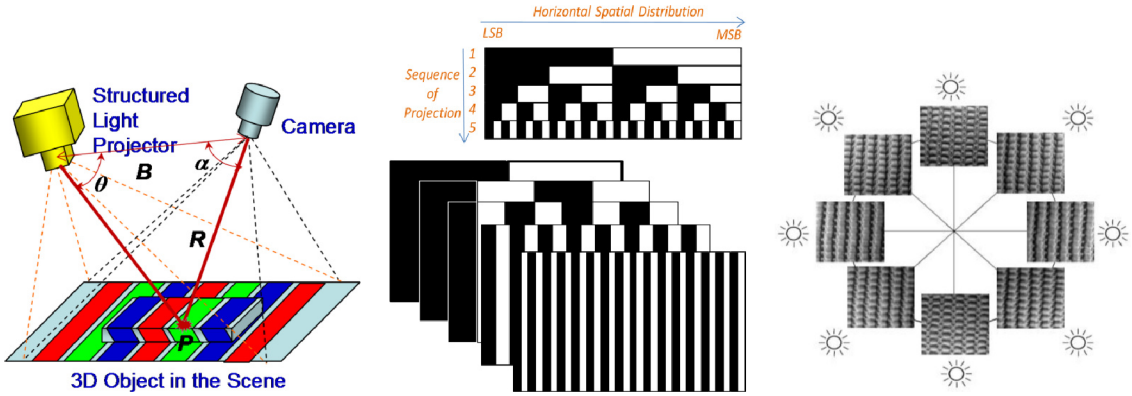
where v_l is the speed of light and t denotes the time it took for one beam of light to travel from the sensor to the target and back. This approach can be used for longer ranges, offers a high reading frequency, and is safe for the eyes, contrary to laser scanning. [1]

Having multiple sensors nearby could cause interference. [2] In the context of 3D face reconstruction and recognition, Apple has been using ToF sensors in their front-facing cameras for several years, which signals the potential of this technology.

Structured-light 3D scanning is a technique used to measure the three-dimensional shape of objects by projecting light patterns onto their surfaces. In this method, a scanner head emits light patterns onto the target object, and cameras capture images of the distorted patterns. By analyzing these distortions, the 3D scanning software reconstructs the object's geometry [5]. Multiple different patterns can be used to illuminate the target surface. Photometric stereo, for instance, uses a sequence of images of the same surface but illuminated from different angles. Binary encoding projects black-and-white patterns, assigning a unique binary code to each pixel [3].

[1] <https://www.terabee.com/time-of-flight-principle/>

[2] <https://www.framos.com/en/articles/advantages-and-disadvantages-of-time-of-flight-cameras>



(a) Structured light (b) Binary encoding (c) Photometric stereo

Figure 1.2: Illustrations of various 3D scanning techniques

Figures 1.2(a), 1.2(b), 1.2(c) are adapted from *Jason Geng, „Structured-light 3D surface imaging: a tutorial," Adv. Opt. Photon. 3, 128-160 (2011)* [5].
[3]

Stereo Vision is a passive technique that uses two or more cameras to capture images of the same scene from slightly different angles. The key idea is based on the concept of parallax, where an object's position appears to shift relative to the background when viewed from different angles.

1. **Image Acquisition:** Two or more cameras capture images of the same scene from different perspectives.
2. **Feature Matching:** Corresponding features (such as edges, corners, or textures) are identified in both images using feature extraction algorithms.
3. **Disparity Calculation:** The horizontal displacement between the matching features, known as disparity, is calculated.
4. **Depth Calculation:** By applying triangulation principles, the depth information of each feature is determined based on the disparity and the known baseline (distance between cameras).

Stereo vision provides depth maps or 3D reconstructions of the scene and is commonly used in applications such as robot navigation, 3D modeling, and autonomous vehicles. However, it requires well-textured surfaces for accurate feature matching and is sensitive to lighting conditions.

[3] <https://www.e-consystems.com/blog/camera/technology/how-time-of-flight-tof-compares-with-other-3d-depth-mapping-technologies/>

Chapter 2

Previous work on 3D face reconstruction and expression modification

In this chapter, I'd like to review the history of 3D face modelling and reconstruction. It's important to acknowledge the predecessors of current state-of-the-art technologies, as each of them laid a foundation for the next improved technique.

I'll start by discussing the **3DMMs** (3D Morphable Models), which is a statistical approach that represents faces in a **parametrized form, where parameters control aspects like shape and texture**. The model can be manipulated by adjusting these parameters, allowing for generating a wide range of faces. 3DMMs have been a foundation in the field of computer graphics for 3D modelling and reconstruction of human faces.

Newer methods utilize artificial neural networks to generate realistic faces and expressions of the human face. One of these more recent models is DECA (Detailed Expression Capture and Analysis), which leverages modern neural network techniques to achieve hyper-realistic 3D reconstruction from flat images and animation of facial gestures regarding person-specific details, such as wrinkles.

2.1 3D Morphable Models (3DMM)

The concept of 3D Morphable Models (3DMMs) was first introduced by Volker Blanz and Thomas Vetter in their paper „A Morphable Model for the Synthesis of 3D Faces“ [2] in 1999. Their pioneering work was presented at SIGGRAPH, which is one of the most prestigious conferences in the field of computer graphics.

Their model allowed for the creation of three-dimensional facial structures from a single two-dimensional image, capturing shape and texture. At the time, this was a substantial advancement which led to others improving upon their work and the paper being cited over 6225 times at the time of writing (december 2023)

2.1.1 Dataset

The dataset used by Vetter and Blanz consists of 200 three-dimensional laser scans (100 male and 100 female) of human faces. The first problem they faced was finding corresponding

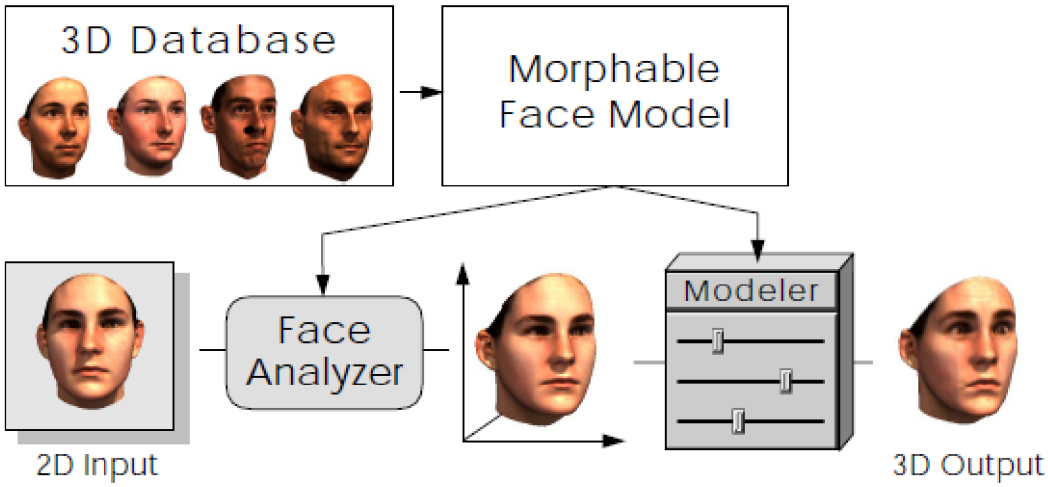


Figure 2.1: 3D Morphable Models adapted from *A Morphable Model For The Synthesis Of 3D Faces* [2]

feature locations in different faces. The correspondence problem is crucial for all morphing techniques, both for the application of motion-capture data to picture or 4D face models and for most 3D face reconstruction techniques from images.

What is the correspondence problem?

The correspondence problem in 3D face modeling involves identifying points or features on one face corresponding to similar points or features on another face. For instance, the tip of the nose on one scan should correspond to the tip of the nose on another.

In the context of 3DMM, establishing a correspondent dataset is crucial for creating a statistical model that can represent a wide range of faces. To establish correspondence, Blanz and Wetter used a technique called **Dense Correspondence**. All 3D scans were aligned to a reference face model in this process. This alignment ensures, that each vertex in the model corresponds to the same anatomical point across all faces.

2.1.2 Parametrization of Face Space

Once correspondence is established, faces can be represented as points in a high-dimensional space. Each axis corresponds to a specific variation in facial features. The shape is represented as a mesh of 3D vertices (x, y and z coordinates for each point). The texture is represented by colour and intensity values for each vertex. In mathematical terms, the geometry of a face is described as a shape-vector

$$S = (X_1, Y_1, Z_1, \ldots, X_n, Y_n, Z_n)^T \in \mathbb{R}^{3n} \tag{2.1}$$

that contains the X, Y and Z coordinates of its n vertices. t

Texture of a face is defined as

$$T = (R_1, G_1, B_1, R_2, \ldots, G_n, B_n)^T \in \mathbb{R}^{3n} \tag{2.2}$$

which contains the red, green and blue value for each vertex.

Face model representation with 3DMM

The morphable model is defined as the set of faces $S_{mod}(\vec{a}), S_{mod}(\vec{b})$, parametrized by the coefficients $\vec{a} = a_1, a_2...a_m{}^T$ and $\vec{b} = b_1, b_2...b_m{}^T$. Arbitrary new faces can be generated by varying the parameters $\vec{a}$ and $\vec{b}$ that control shape and texture. In theory, this allows for creating every possible face by fine-tuning these parameters.

Principle Component Analysis (PCA)

Authors of the paper then use Principal Component Analysis (PCA), which is commonly used for reducing dimensionality of a dataset while retaining as much of the variability in the data as possible.
This process involves first calculating the covariance matrices C_S and C_S computer over the shape and texture differences

$$\Delta S_i = S_i - \bar{S} \tag{2.3}$$

$$\Delta T_i = T_i - \bar{T} \tag{2.4}$$

where S_i, T_i represents the shape and texture of the i_{th} sample, respectively. $\bar{S}$ and $\bar{T}$ represent the average shape and texture of all faces.
Covariance matrices are sorted in descending order according to their eigenvalues - the higher the eigenvalue, the greater of an impact a given eigenvector has on the result. This means that the first few eigenvectors capture the most significant patterns of variation in the dataset. New coordinate system is formed using the eigenvectors s_i and t_i calculated from the covariance matrices.

2.1.3 Changing facial expressions

The paper by Vetter and Blanz describes changing facial expressions in a morphable face model. They differentiate between dynamic facial expressions, which can be transferred by recording change from a neutral state, and static attributes like gender or nose shape. For static attributes, they use a set of labelled faces to define shape and texture vectors that, when adjusted, can manipulate a specific attribute while keeping others constant. This approach provides a method to model and manipulate both dynamic expressions and static facial attributes.

2.2 Blendshape models

The following section is inspired by Practice and Theory of Blendshape Facial Models „ [12]

Blendshapes are linear facial models in which the individual basis vectors are not orthogonal but instead represent individual facial expressions. The corresponding **weights** are often called **sliders**, since this is how they appear in the user interface. In contrast to morphable models, described in 2, the underlying basis of blendshapes is semantic rather than orthogonal.

Semantic representation refers to each vector in the basis representing a specific, meaningful dimension or feature within the data. In the context of blendshapes, these could represent a frown, or a smile, etc.

2.2.1 Practical applications of blendshapes

Blendshape models have revolutionized the **movie industry** by providing a powerful tool for creating realistic and expressive facial animations. They have been successfully employed in many popular movies, including **Avatar and The Lord of the Rings**.

In the **gaming industry**, blendshapes are used for animating characters, enabling them do display a wide range of emotions and reactions that enhance the gaming experience. Modern game engines like **Unity or Unreal Engine** support the use of blendshapes.

Drawbacks The creation of a blendshape is very labour and cost-expensive, primarily because it requires the expertise of a skilled modelling artist to craft each necessary facial expression.

2.3 Photorealistic and Articulated Computational face model (PHACE)

PHACE is a cutting-edge physics-based modelling technique which leverages advanced simulations to replicate the intricate anatomy of the human face. This innovative approach allows for the accurate prediction of the dynamic forces applied to individual muscles of the face. This technique differs from other methods, as it very closely resembles the actual structures of the face, simulating interactions between facial components, like rigid bones structures, active muscle tissues and passive flesh, fat, and skin layers in a fully volumetric simulation model of the human face. [9]

2.3.1 Expression modification

Various deformations of the active regions - muscles, can be programatically controlled in order to simulate biomechanical muscle contractions. PHACE allows for rotation and translation of the head model and repositioning of the jawbone. The average male head comes with 48 blendshapes. Interestingly, the teeth are replaced with simplified meshes due to their complex geometry. To better model facial expressions, the skin is allowed to slide along the bones in some regions. PHACE achieves heightened realism through advanced collision handling mechanisms that authentically simulate interactions within facial tissues and bones, as well as with external elements like wind or virtual reality headsets. The simulation accurately captures internal and external collisions, enhancing the overall authenticity of facial expressions and movements.

2.4 Detailed Expression Capture and Animation (DECA)

In recent years, machine learning and neural networks have become increasingly popular. These advancements have paved their way into various fields, including 3D face reconstruction and expression modification.

DECA is an advanced method for reconstructing and animating detailed 3D facial models, particularly from single images. Developed to address the limitations of previous 3D

face reconstruction methods, DECA provides a comprehensive approach for capturing both coarse structure and fine-scale details of human faces, including intricate facial expressions. [4]

The **key idea** of DECA is grounded in the observation that an individual's face shows different details (i.e. wrinkles), depending on their facial expressions but that other properties of their shape remain unchanged. Consequently, facial details should be separated into static person-specific details and dynamic expression-dependent details such as wrinkles [4]

Coarse structure

The term "coarse structure„ refers to the fundamental underlying shape - in our context, the shape of a human face - excluding finer details, which includes the overall facial geometry and proportions, such as the shape and size of the nose, eyes, jawline and other facial contours.

2.4.1 FLAME (Faces Learned with an Articulated Model and Expressions)

DECA uses FLAME [13] as a fundamental framework for modelling and animating 3D faces. FLAME uses a set of parameters to define the shape, expression, and pose of a 3D face model. These parameters control various aspects of the facial structure and movements.

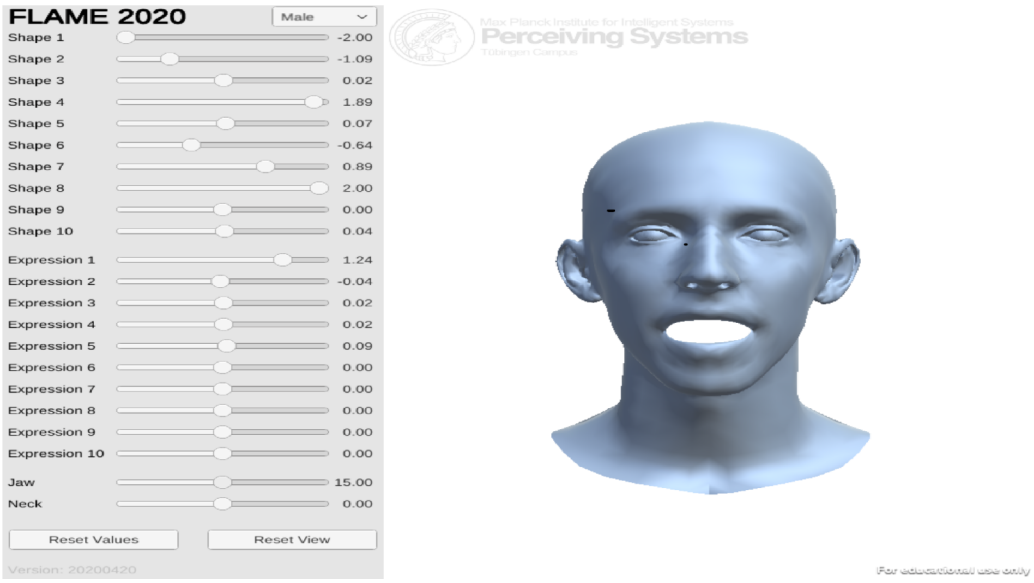


Figure 2.2: Online FLAME editor for changing shape and expressions

Figure 2.3 depicts an online editor powered by FLAME. It visualizes the first few parameters for shapes and expressions. By tweaking the shape sliders, various parts of the face are modified in real-time, e.g. height and width of the face, length of the neck or eye width. Modifying expression sliders results in the model expressing different emotions - smiling, frowning, etc. FLAME also allows for changing the head's orientation (tilt, pan).

[0] <https://flame.is.tue.mpg.de/interactivemodelviewer.html>

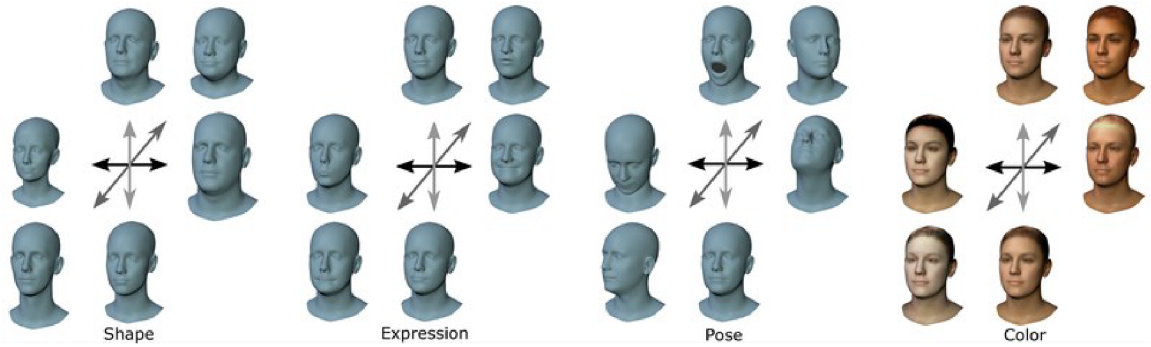


Figure 2.3: FLAME model variations for shape, expression, pose, and appearance. The first three principal components are visualized at ±3 standard deviations for shape, expression, and appearance variations. The pose variations are visualized at $\pi/6$ (head pose) and 0,$\pi/8$ (jaw articulation). [13]

2.4.2 Representation of a 3D model

DECA builds upon FLAME by adding more detailed, expression-dependent facial features such as wrinkles, or texture changes, which are not captured by FLAME alone. While FLAME provides a strong foundation for the overall shape and basic expressions, DECA adds high-resolution details, like wrinkles, subtle skin texture changes, and detailed expression dynamics. DECA focuses on expression-dependent details that can change with facial expressions. For example, the appearance of wrinkles when smiling or frowning is captured more accurately in DECA, a level of detail not provided by FLAME.

2.4.3 Training

Contrary to most 3D face reconstruction methods, DECA uses in-the-wild training images, meaning samples captured in natural, uncontrolled environments instead of those taken in controlled settings like studious. This allows for greater generalization.

2.4.4 Coarse reconstruction

DECA uses an encoder-decoder architecture to reconstruct coarse shapes. Given a 2D image as input, it is encoded into a latent code. An encoder, which consists of ResNet50 [7] network is trained on 2D input images, followed by a fully connected layer to regress a low-dimensional latent code consisting of FLAME parameters representing coarse geometry. The latent code serves as an input to a decoder, which synthesises a 2D image.

The encoder-decoder architecture is ideal for reducing data dimensionality. In this case, the aim is to encode a 2D image, typically represented as a flattened vector, into a compact, low-dimensional latent code and then reconstruct using a decoder.

As outlined in the DECA paper [4], the latent code comprises FLAME parameters [13] β, Ψ, Θ, describing the **shape, expression and pose**. Additionally, DECA incorporates albedo [1] coefficients α, camera c, and lighting parameters I.

RingNet, a similar technique [22] also utilizes ResNet50 and regresses into latent code, but differs in detail reconstruction. There are a total of 236 parameters, which predict the latent code. [1]

Calculating loss

There are multiple losses that DECA uses to estimate the accuracy of the model. The loss function measures the discrepancies between the original image and the reconstructed image. The lower the loss, the greater the resemblance between the original and reconstructed image.

To calculate the coarse reconstruction loss, we use the following formula: [4]

$$L_{coarse} = L_{lmk} + L_{eye} + L_{pho} + L_{id} = L_{sc} + L_{reg} \tag{2.5}$$

L_{lmk} **- Landmark re-projection loss**
L_{eye} **- Eye closure loss**
L_{pho} **- Photometric loss**
L_{id} **- Identity loss**
L_{sc} **- Shape consistency loss**
L_{reg} **- Regularization**

2.4.5 Detail Reconstruction

DECA enhances the base model with high-resolution details using a specialized **detail network** that captures expression-dependent wrinkles and other fine-scale features.

- **Displacement Map:** DECA introduces a displacement map to capture high-frequency facial details not represented in the FLAME model. This map is added to the FLAME mesh, enriching it with finer facial features.
- **Detail Network Architecture:**
  - **Input:** The detail network takes a 2D image as input.
  - **Encoder-Decoder Structure:** Uses an encoder-decoder structure to predict a displacement map. The encoder extracts features, and the decoder regresses them into a high-resolution displacement map.
  - **Output:** The displacement map is added to the FLAME mesh.

The final architecture combines the coarse FLAME model with the detail network to produce highly detailed 3D reconstructions.

[1] Albedo is a measure of how reflective a surface is. <https://www.metoffice.gov.uk/weather/learn-about/weather/atmosphere/albedo>.

Chapter 3

Proposed approach for 3D face reconstruction and expression modification

In this chapter, I'll propose an approach for reconstructing a 3D face model from a single 2D image using unsupervised learning. The architecture is based around FLAME model, described in 2.3. Contrary to other 3D reconstruction methods, my approach will only reconstruct the coarse shape without texture or albedo.

Overview of the Process First, we must acquire a dataset of face images, which ideally have the same format and dimensions. We then need to figure out a way to extract landmarks from a face in a differentiable way. The input dataset will comprise an image and its extracted landmarks. After obtaining the images and landmarks, we can pass the input images through a network which regresses into a latent code consisting of FLAME [13] parameters. To reconstruct a 3D model from these parameters, we can use a modified Tensorflow [1] implementation of FLAME [1] tailored to our needs. After creating a 3D model, we need a differentiable renderer to project an object to 2D. Finally, we extract facial landmarks from the rasterized image and using an adequate loss function, compare them with the original landmarks.

3.1 Dataset

For the dataset, the Chicago Face Database (CFD) [18] [17] [16] was chosen (also used in 5). The input to the network consists of aligned, high-resolution images of human faces. There are several options for facial landmark detection. One of the most popular solutions is dlib [2]. To ensure that we can calculate derivatives of the output with respect to our initial model parameters, our solution for landmark detection has to be differentiable.

One solution is built around the LaPa [14] dataset, which contains 22000 images, each with 106-point landmarks. This author of this repository on GitHub [3] utilized the dataset and trained a Tensorflow model to predict the positions of these landmarks. The dataset contains 831 images of faces and their corresponding landmarks.

[1] https://github.com/TimoBolkart/TF_FLAME.git
[2] https://github.com/davisking/dlib
[3] https://github.com/nikhilroxtomar/Human-Face-Landmark-Detection-in-TensorFlow

3.2 3D Model Generation

The input image is fed through a neural network that regresses into FLAME parameters for translation, rotation, pose, shape and expressions. These parameters are stacked onto each other in the output layer, forming a single 1D array.

We can leverage the TensorFlow implementation of FLAME to create a point cloud object characterized by **vertices**, each comprising x, y, and z coordinates, as well as **faces** that establish connections between these vertices. This point cloud object can then be seamlessly integrated with the SMPL model [15], enabling the SMPL model to process and manipulate the generated geometry effectively.

3.3 Converting a 3D Object to 2D

To detect landmarks on our predicted model, we need to convert our 3D generated face into 2D. The process involves these steps

1. **Projection** - The first step is projecting 3D vertices onto a 2D plane using projection techniques (like perspective or orthographic projection)

2. **Clipping** - Involved removing parts of the geometry that lie outside the viewable are to optimize rendering. (In our case this might not be necessary)

3. **Rasterization** - In this stage, the projected 2D triangles are converted into a set of pixels on the 2D screen.

When projecting a 3D face onto a 2D plane, it is essential to ensure that the process remains differentiable so it can seamlessly integrate into our existing architecture. Several differentiable renderers are commonly used in machine learning:

- **Tensorflow Graphics** This library includes differentiable layers that are easily integrated with TensorFlow [19]

- **tf_mesh_renderer** a 3D mesh renderer implemented in TensorFlow, as detailed in [6]

- **DIRT: a fast Differentiable Renderer for TensorFlow** - A fast renderer that employs OpenGL for rasterization while offering support for GPU and CUDA [8]

After countless failed attempts at making these solutions work with Tensorflow.2x, which is a requirement given other parts of the code, I found one suitable for this problem.

pTFrenderer [23], which stands for "Pure TensorFlow Renderer„, is a "differentiable renderer written purely in TensorFlow, which is capable of rendering triangle meshes„. [4]

[4] https://github.com/szattila/pTFrenderer

3.4 Landmark Detection and Loss Calculation

After projecting the face model onto a 2D image, we can use the same landmark detection process we used to create the training dataset. After identifying the landmarks in the predicted image, I utilized a mean squared loss function to measure the disparity between these predicted landmarks and the ground truth landmarks extracted from the input image.
Since the entire process, including landmark detection, generating 3D models from parameters, and mesh rendering, is designed to be differentiable, we can calculate gradients of the loss function with respect to the model's parameters. This enables us to optimize the model parameters through backpropagation, allowing the system to learn and improve its ability to reconstruct faces accurately

Chapter 4

Reconstructing a 3D Face Model From a Single Image

4.1 Used Tools

My initial mistake was attempting to code and execute everything on a Windows computer. Recognizing the limitations and challenges of running machine learning-related tasks on a Windows environment, I transitioned to a Linux-based machine, which offered better compatibility and support for the project requirements. Ultimately, I found Google Colab to be a more suitable platform, providing the necessary resources and infrastructure for seamless development and execution of machine learning solutions. I decided to utilize the Google Colab notebook [1] which installs all necessary requirements and showcases the individual parts of my solution. Other helper functions and models can be found at my GitHub repository. [2]. GitHub links in footnotes were used as sources for my solution.

I'll start by showcasing interesting parts of the code in the order described in the previous chapter 3 and then combine the pieces.

4.2 Dataset Creation

To create the input dataset, the function `create_dataset()` located in **build_utils.py** is utilized;

```
# simplified version of the create_dataset function
def create_dataset(path):
    dataset = {'images': [], 'landmarks': []}
    for file in path:
        img = load_img(file)
        preprocess_img(img)
        lmks = detect_landmarks(img)
        dataset['images'].append(img); dataset['landmarks'].append(lmks)

    return dataset
```

[1] <https://colab.research.google.com/drive/1sa5BdaoV2XcoyRXvVSsla-ANzVkuvb04#scrollTo=eQtfooixyUzi&uniqifier=2>

[2] <https://github.com/Pojzo/3D-face-reconstruction-using-FLAME>

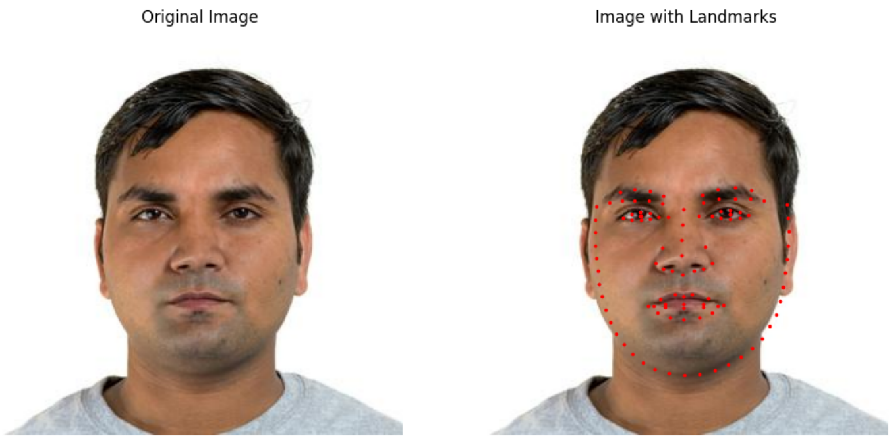


Figure 4.1: Input landmarks sample

We can visualize the detected landmarks using the `visualize_landmarks()` function in **render_utils.py**. The shape of image is $(224, 224, 3)$ - height, width, n_channels. We need to supply the path to the pre-trained landmark detection model to predict the landmarks, load it, and supply an input image. The prediction is represented as a 1D tensor of length 212.

```
# simplified version of landmark detection
def detect_landmarks(lmks_model_path, input_image):
    lmks_model = load_model(lmks_model_path)
    pred_landmarks = lmks_model(input_image)

    # the format of the output is
    # [lmk_x1, lmk_y1, lmk_x2, lmk_y2...]

    xs = pred_landmarks[::2]
    ys = pred_landmarks[1::2]

    return xs, ys
```

4.3 Processing the predicted face

As mentioned in previous parts, the FLAME models consists of 5 sets of parameters, which are predicted by our model. The output is a 1D 418 long Tensor that needs to be converted to specific parameters. We can extract the flame parameters by using a helper function `extract_flame_params()` defined in **my_utils.py**.

4.3.1 Extracting Flame Parameters

```
def extract_flame_params(data: tf.Tensor) -> Dict[str, tf.Tensor]:
params = {
    'trans': data[..., :3 ],
    'rot' : data[..., 3:6 ],
    'pose' : data[..., 6:18 ],
    'shape': data[..., 18:318],
    'exp' : data[..., 318: ]
}

return params
```

4.3.2 Reconstructing a 3D model and subsequent rendering

Now that we have the predicted parameters, we will use a modified version of the Tensorflow FLAME implementation, with the help of the SMPL model to generate a 3D representation, which includes 5023 points (x, y, z) and a set of faces.

```
# this function is adapted from the TF_FLAME repository
def render_face_from_flame(flame_params: Dict[str, tf.Tensor], smpl):
    tf_trans = flame_params['trans']
    tf_shape = flame_params['shape']
    tf_exp = flame_params['exp']
    tf_rot = flame_params['rot']
    tf_pose = flame_params['pose']

    # get the vertices - reconstruction from parameters
    vertices = tf.squeeze(smpl(tf_trans, tf.concat([tf_shape, tf_exp], axis=-1), tf.con

    # render a 2d image using a differentiable renderer
    rendered_image = render_image_from_mesh(vertices, faces)

    # return the rendered image + 3D model (vertices + faces)
    return rendered_image, (vertices, faces)
```

Figure 4.2 shows the reconstructed image (vertices, faces) and image rendered using a differentiable renderer

Figure 4.2: Comparison: right - Reconstructed 3D face; left - rendered face using a differentiable renderer

4.3.3 Landmark prediction on the output image

Now that we have a rendered face, we can use the same landmark detection model used to create our dataset of input landmarks.

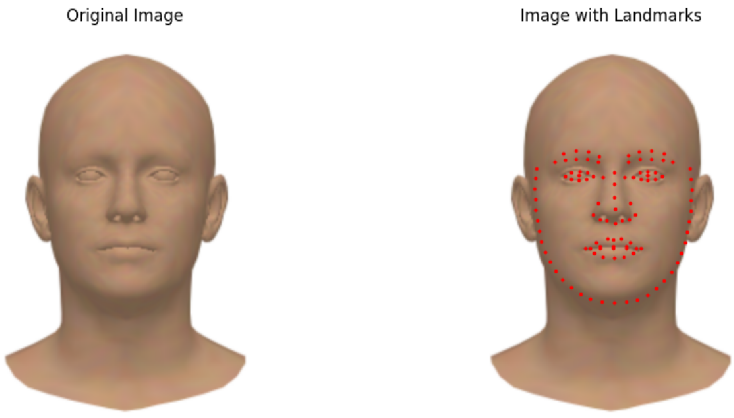


Figure 4.3: Output landmarks sample, same as 4.1 but with predicted face

4.4 Evaluation example

To evaluate the results of the training process, we need to use a suitable loss function. In this case, we can use Mean Squared Error (MSE) to compare the input landmarks with the output landmarks. The MSE loss is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

where N is the total number of landmarks, y_i represents the ground truth landmarks, and $\hat{y}_i$ denotes the predicted landmarks. Using this loss function, we can effectively measure the similarity between the predicted and actual landmark coordinates.

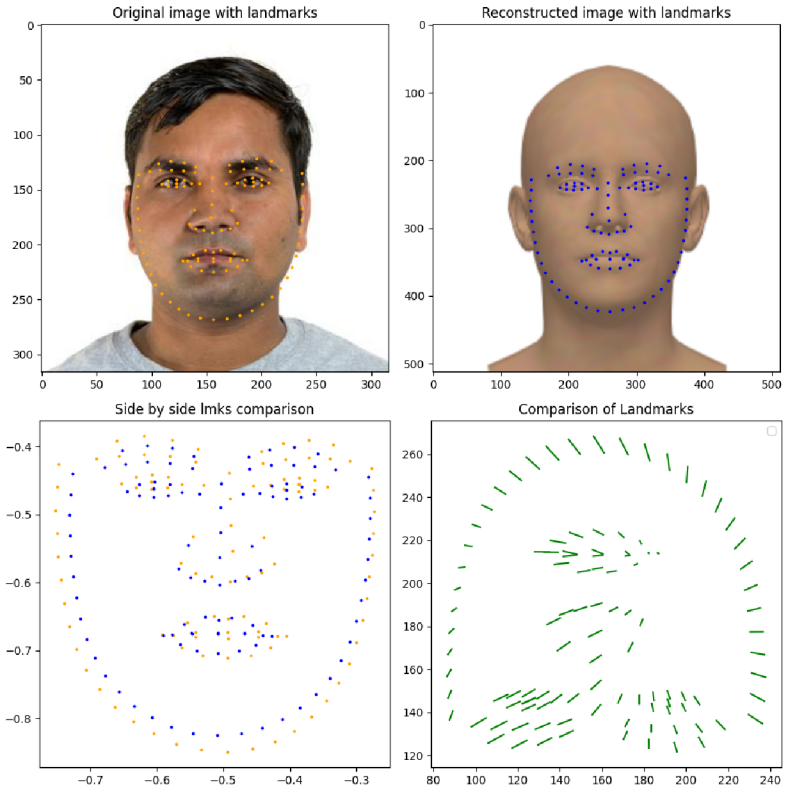
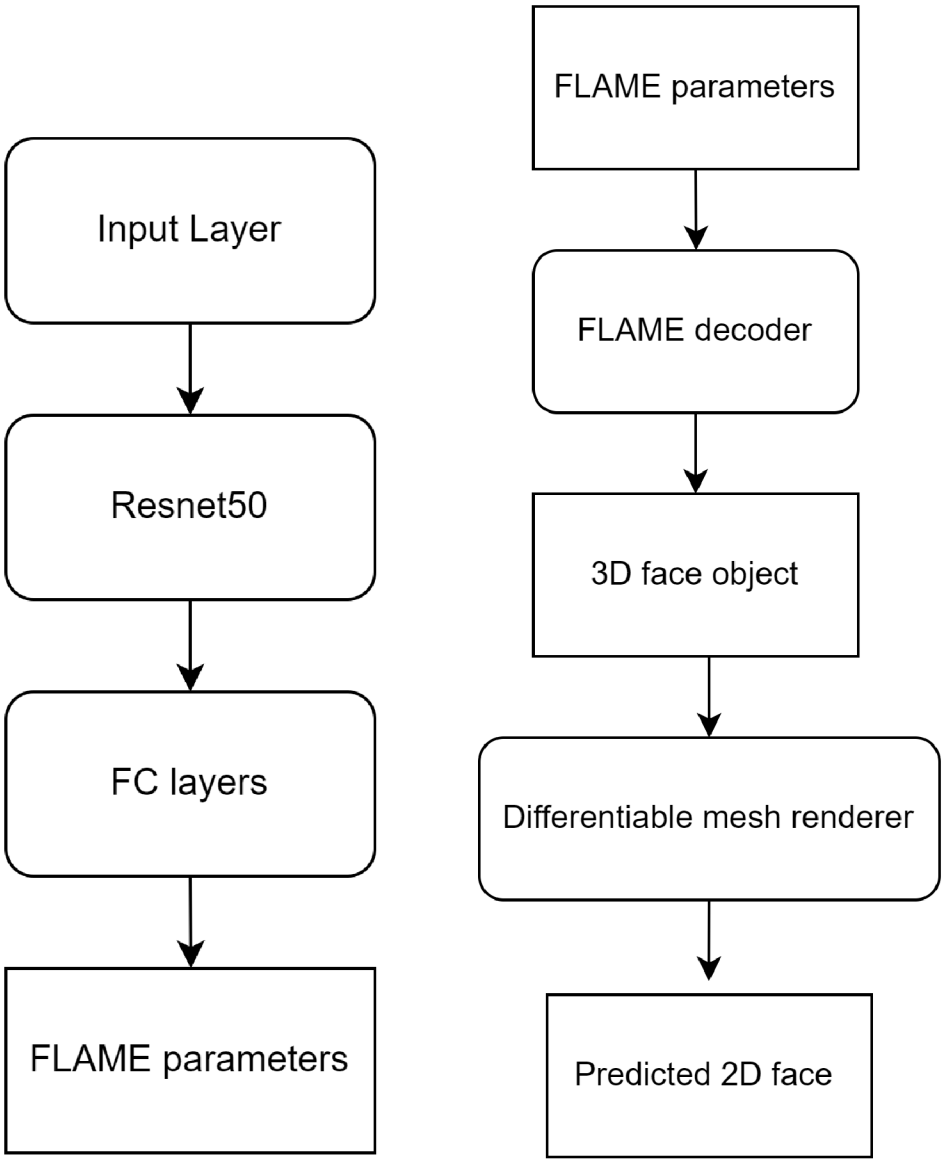


Figure 4.4: Comparison of input and predicted facial landmarks. Top-left: Original image with landmarks. Top-right: Reconstructed image with landmarks. Bottom-left: Side-by-side comparison of normalized landmarks. Bottom-right: Landmark position shifts visualized with arrows.

The figure 4.4 was generated using the `visualize_input_output_landmarks()` function.

4.5 Putting the pieces together

The network architecture is as follows: 2D image on the input goes through a pre-trained ResNet50 model [7], followed by a few fully connected layers that regress into FLAME parameters. For simplicity purposes, all weights except shape, which has the most influence on the resulting object are set to `trainable=False`. FLAME decoder is then used to contruct a 3D model from the latent parameters.



(a) Neutral Network architecture - Part 1 (b) Neutral Network architecture - Part 2

Figure 4.5: Model architecture - round objects represent data flow, edged objects represent data. Code for the left part is below this figure.

```
class FLAMEModel(Model):
    def __init__(self):
        super(FLAMEModel, self).__init__()
        base_model = tf.keras.applications.ResNet50()
        base_model.trainable = False

        self.encoder = Model(inputs=base_model.input, outputs=base_model.output)

        # Fully connected layers
        self.fc1 = layers.Dense(1024, activation='relu')
        self.fc2 = layers.Dense(512, activation='relu')

        # Custom Output Layer for FLAME parameters
        self.output_layer = OutputLayer()
```

Now to put it all together, we have a dataset with images and their corresponding landmarks. We have a model that can predict FLAME parameters. FLAME decoder which creates a 3D model. For training purposes, a differentiable renderer is used to project the generated 3D object to 2D. Landmarks are extracted

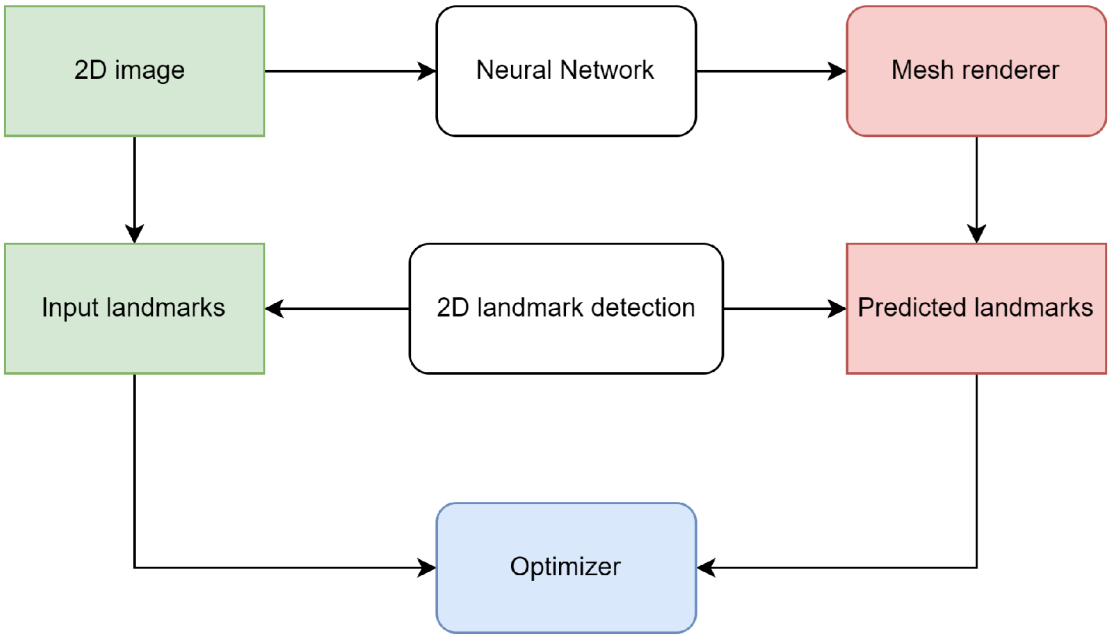


Figure 4.6: Flowchart representing the whole process of generating a mesh, projecting and extracting predicting landmarks, omitting the details present in the figures 4.5(a) and 4.5(b). Mesh renderer represents the predicted projected 2D image

4.5.1 Training

```
def train():
    # simplified version of the training
    for epoch in range(epochs):
        # assume device GPU:0
        input_image, input_landmarks = dataset['images'], dataset['landmarks']
        flame_params = process_output(model(input_image))

        model_3d = flame_generate_face(flame_params)
        projected_2d_image = render_from_flame(model_3d)
        pred_landmarks = extract_landmarks(projected_2d_image)

        loss = calculate_loss(input_landmarks, pred_landmarks)
        gradients = calculate_gradients(loss, model)
        model.backward(gradients)
```

4.6 Process for Expression Modification

The FLAME model has a set of parameters specifically dedicated to facial expressions. These parameters can modify aspects like smiling, frowning, or surprise. Adjusting the indices corresponding to facial expressions allows for changes in expressions such as raising an eyebrow, smiling, or squinting. For instance, increasing the parameter associated with smiling will produce a smiling expression on the 3D face.

4.7 Challenges

The objective of my bachelor's thesis was to develop a solution that can reconstruct a 3D face from a 2D image. My proposed approach involved using a neural network architecture, starting with a pre-trained ResNet50 model, followed by fully connected layers, and finally predicting FLAME parameters for the reconstruction. Even though the loss consistently decreases during training, the resemblance of the reconstructed 3D model to the input image is often minimal or appears random. The model seems to have learned to manipulate the landmark detection algorithm rather than learning the proper structure of the FLAME parameters. This is likely because the landmark data provides insufficient guidance for accurate 3D face reconstruction.

4.7.1 Proposed Refinements

3D Landmarks

Instead of using 2D landmarks, which may be insufficient, we can directly extract landmarks from the 3D generated face. These 3D landmarks can then be projected to 2D using, for example, a weak perspective, which is used by DECA [4] and then comparing them with the ground truth.

Photometric and Identity-Based Losses

Adding photometric loss by comparing pixel values between the input image nd the 3D model can provide more comprehensive guidance. Incorporating identity-based losses using pre-trained face recognition models will help the network capture consistent facial shapes.

4.7.2 Additional Enhancements

Incorporating additional data sources like texture maps and facial segmentation could improve the reconstruction accuracy

Chapter 5

Studying the impact of expression modification on 3D face recognition

The goal of the final part of my bachelor's thesis was to generate multiple expressions and study the modified face's impact on 3D face recognition. I used the state-of-the-art solution [4] to reconstruct faces from 2D images. As for the input for the reconstruction, I chose to use the Chicago Face Dataset (CFD) [18], as it contains a large number of quality, well-lit, multi-racial images displaying human faces with neutral expressions.

After a face model is reconstructed, a set of expressions is transferred onto the neutral face using DECA's expression transfer tool. One sample of the resulting dataset comprises a neutral 3D model and n models of the same person showing an expression.

Looking for a 3D face recognition solution that accepts directly three-dimensional data as input proved more difficult than anticipated. My first thought was to look for a Python module aimed at such a problem. After an unsuccessful search, I tried looking for a solution in a 2022 survey on 3D face recognition [11]. The article compares various studies, their methodologies, advantages, limitations, and evaluation results of standardized metrics. The main caveat of most existing solutions is that they use 2D-input networks, meaning that 3D faces are converted to 2D maps as input for the network. A few, such as PointNet++ [20] or Pointface [10], can directly handle three-dimensional data as inputs to the network. Given that it's a requirement for the topic of this thesis, only these methods can be discussed. After some research, I figured Pointnet++ is a suitable option. One disadvantage of this study is that it's utilized on everyday objects, not human faces.

To address this issue, researchers have adapted Pointnet's architecture to better suit the recognition of human faces in a study titled "Learning Directly from Synthetic Point Clouds for 'In-the-wild' 3D Face Recognition„ [25]. This method performs almost perfect scores on evaluation metrics. The recognition works by supplying Euclidean coordinates (x, y, z), normals (nx, ny, nz) and surface curvature. After attempting to use the pre-trained model on my data, a related problem emerged, which prevented further use of the model. Upon writing an email to a researcher who forked the GitHub repository, I found out he experienced the same issues. Due to problems with the repository, I decided to use PointNeXt [21], a successor to Pointnet++.

5.1 Overview

The whole process of evaluation involves these steps:

- Prepare dataset of neutral faces
- Create a collection of expressions for transfer
- Generate 3D models of neutral faces using DECA
- Transfer emotions from the collection to the neutral faces
- Run experiments using various combinations
- Evaluate results

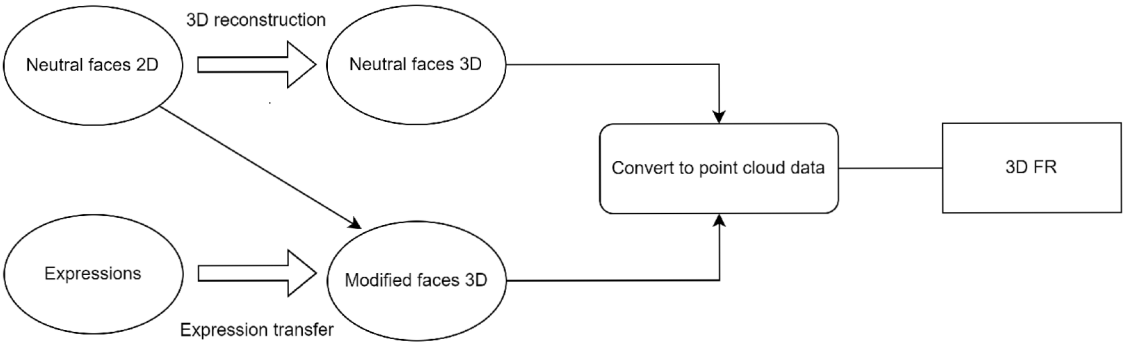


Figure 5.1: Data preparation for 3D recognition system

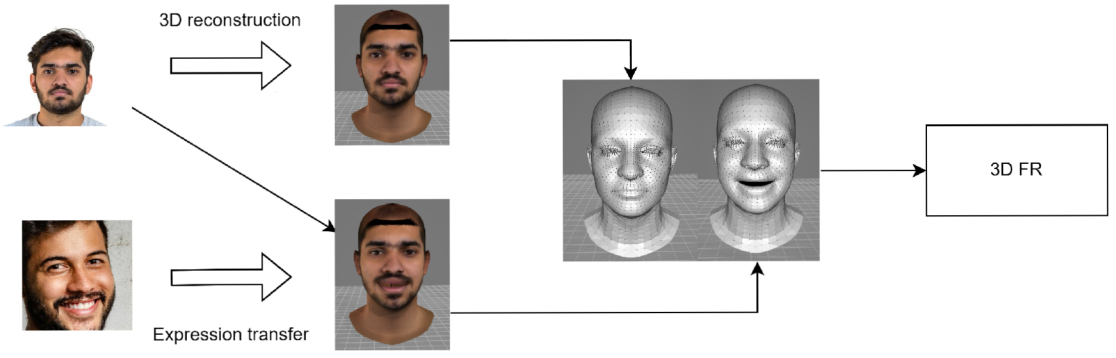


Figure 5.2: Data preparation for 3D recognition system - example

5.2 Selected expressions

A total of 7 different emotions were chosen for the purpose of expression transfer, namely anger, confusion, disgust, fear, happiness, sadness, and surprise. These emotions represent a broad spectrum of human facial expressions, covering both positive and negative feelings. The selection aims to evaluate the robustness of 3D face reconstruction algorithms under various emotional states.

Figure 5.3: Images used as baseline expressions

Figures 5.4, 5.5, 5.6 display a few examples, in order from left to right: neutral 2D face, neutral 3D reconstructed face, and 3D face with transferred expression.

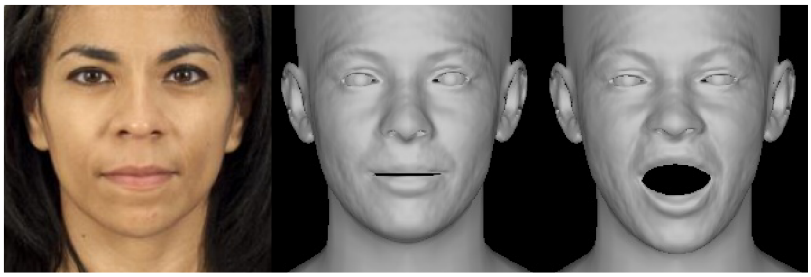


Figure 5.4: Anger

Figure 5.5: Fear

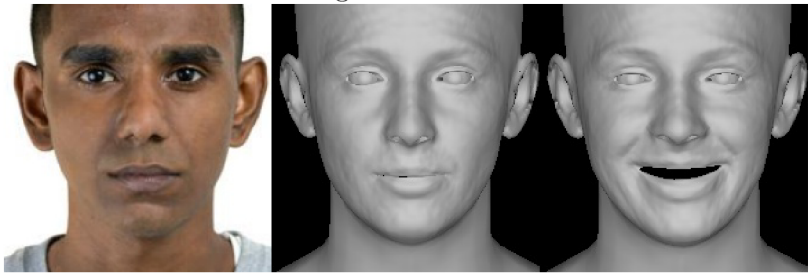


Figure 5.6: Happiness

Figure 5.7: Images representing different emotions 2D -> 3D

5.3 Experiments

Three different kinds of experiments were conducted, and the data will be further denoted as “trainingtesting„ data. For example, “neutral-neutral„ means the model was trained and tested on faces depicting neutral expressions, “neutral-modified„ means the model was trained on neutral faces and tested on modified faces, and “modified-modified„ means the model was trained and validated on faces with both neutral and modified faces.

Multiple input shapes (size of point cloud data) were used to measure the impact of the density on the results. The number of points for the current plot can be seen in the title of the subplot. More precisely, 1024, 2048, 4096 and 8192 sizes for point cloud data were used.

5.3.1 Experiment 1 - using neutral expressions as input to the 3D recognition model

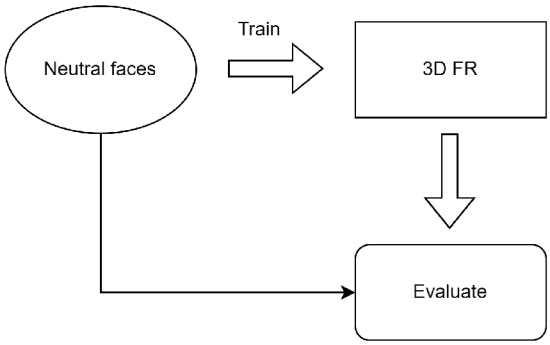
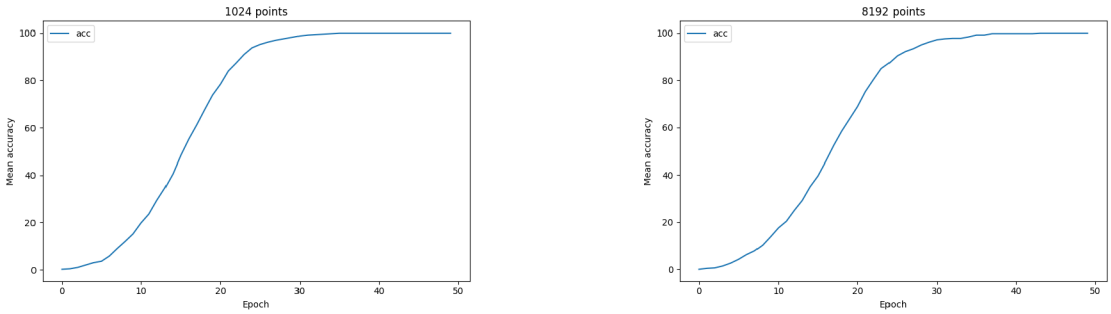


Figure 5.8: Flow chart neutral-neutral

The first idea is to study the accuracy of 3D face recognition using just neutral expressions. The training aims to train the model to categorize a face based on a point cloud derived directly from the reconstructed face model. To account for real-life conditions, input point clouds are manipulated using transformations such as rotation, scaling and randomly moving points. The results of this experiment are shown in the figure below 5.9. The model learned to categorise faces in just 50 epochs for all categories accurately.



(a) neutral-modified 1024 points

(b) neutral-modified 8192 points

Figure 5.9: Results for experiment neutral-modified

5.3.2 Experiment 2 - using neutral expressions as training data and modified expressions as validation data

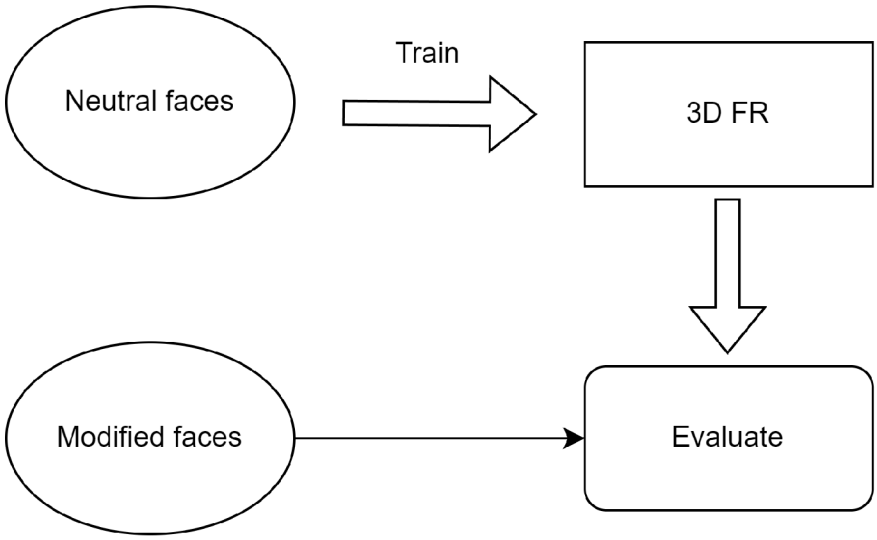
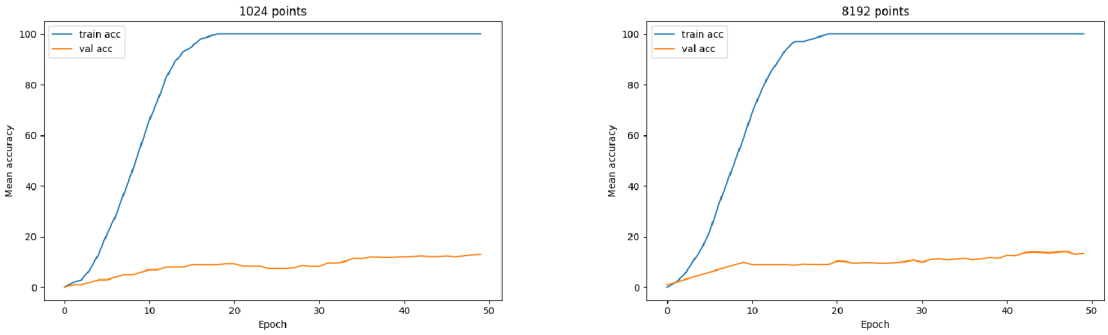


Figure 5.10: Flow chart neutral-modified

The second idea is to use neutral faces for the training and faces with altered expressions as validation data to see if the model can learn to recognize human faces even if their expressions change. The results came out as anticipated, with the model being unable to reach a high level of accuracy. Increasing the number of points, however, seems to have increased the mean accuracy slightly, although it began to stall even at 8192 points. Increasing the number of points would seemingly improve the accuracy even more, although not to the level where the recognition could be used reliably, as the task is too difficult given the limited input data.



(a) neutral-modified 1024 points

(b) neutral-modified 8192 points

Figure 5.11: Results for experiment neutral-modified

5.3.3 Experiment 3 - using neutral and modified faces in training and validation

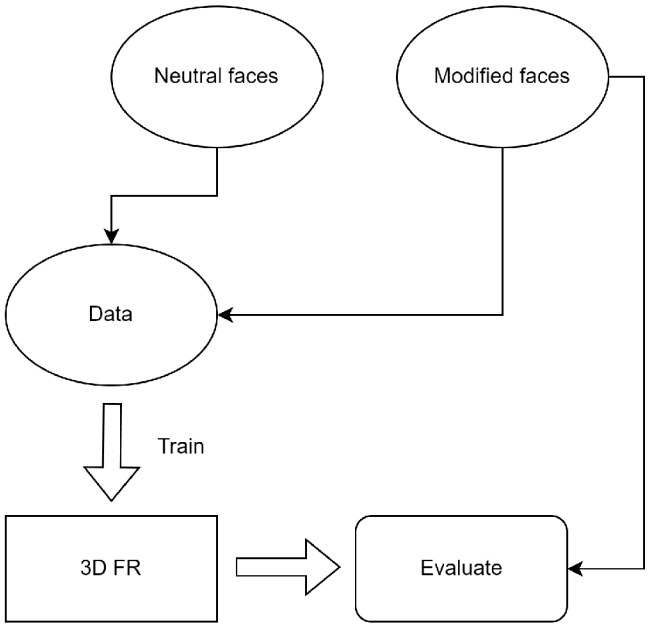


Figure 5.12: Flow chart modified-modified

The third experiment consisted of training the model on both neutral and modified faces to see if using multiple variable samples per person would result in greater accuracy. As hypothesised, the results look promising. Even though it took much longer to train, including more samples/expressions increased the accuracy significantly. It seems that adding more points to the input data results in no further improvement, as can be seen on figure 5.13. The model hit a plateau in all four cases at 60%. Though this is an improvement from the previous experiment, the results are unreliable for real-life scenarios.

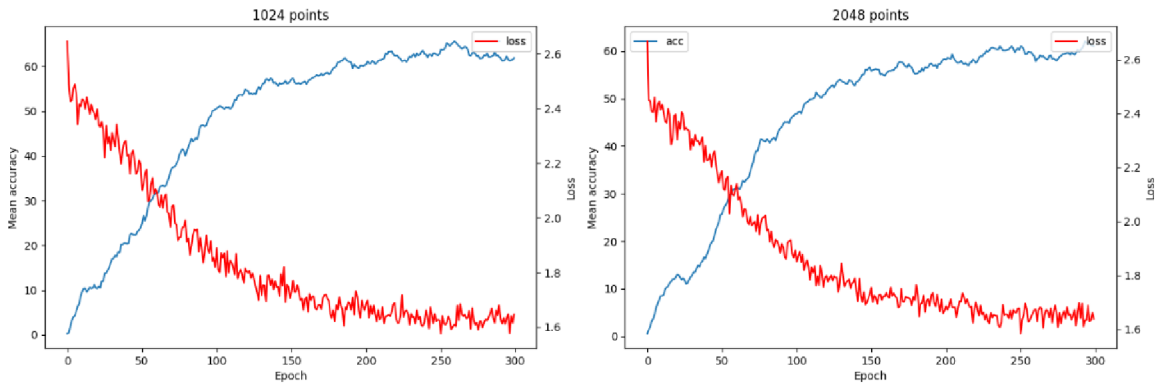


Figure 5.13: Results for experiment modified-modified

5.4 Results

The final results for each experiment are as follows

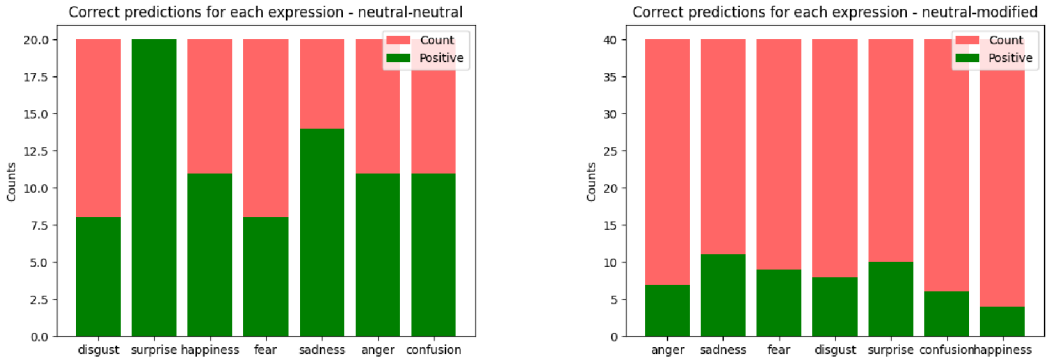
neutral-neutral: Perfect accuracy on 10 and 50 classes after 30 epochs for all input sizes - 1024, 2048, 4096, 8192

neutral-modified: Accuracy of 12% on never-before-seen data of modified faces. Increased input data density resulted in no significant improvements. Given that there were only 10 classes, the predictions of the model are only 2% than if the class was chosen randomly. Using more classes, therefore, would be pointless.

modified-modified: After 300 epochs, the model reached an accuracy of 60% on all input sizes.

5.4.1 Individual expressions

It's interesting to see which expressions the models struggled the most with and which expressions were most likely for the model to predict the correct class.



(a) neutral-neutral

(b) neutral-modified

Figure 5.14: Resulting accuracies for individual expressions

The figure 5.14 clearly show that using more diverse training data results in more accuracy for all expressions. Both figures show high accuracy for surprise and sadness. Other expressions seem not to correlate.

5.5 Proposed enhancements

Given that the third experiment resulted in an accuracy of only 60% improvements are definitely needed. It would be interesting to study different 3D recognition systems and compare them against each other. Perhaps the accuracy is influenced by the chosen expressions, or rather their representative images. Reconstructing 3D models from real faces and transferring emotions is a lengthy and computationally expensive process, so the answer to better results might be to generate a large sample of faces using 3D Morphable Models, as mentioned in 2 for training and real faces for validation. [25]

5.6 Code sources

- <https://github.com/yfeng95/DECA.git>
- <https://github.com/guochengqian/PointNeXt.git>
- <https://github.com/charlesq34/pointnet2.git>
- <https://github.com/szattila/pTFrenderer.git>
- <https://github.com/nikhilroxtomar/Human-Face-Landmark-Detection-in-TensorFlow>
- <https://github.com/guochengqian/openpoints.git>
- <https://github.com/MPI-IS/mesh.git>
- <https://github.com/TimoBolkart/TF_FLAME.git>

Chapter 6

Conclusion

6.1 Chapter 4 Conclusion

My proposed approach involved using a neural network architecture, starting with a pre-trained ResNet50 model, followed by fully connected layers, and finally predicting FLAME parameters for the reconstruction. Even though the loss consistently decreases during training, the resemblance of the reconstructed 3D model to the input image is often minimal or appears random. The model seems to have learned to manipulate the landmark detection algorithm rather than learning the proper structure of the FLAME parameters. This is likely because the landmark data provides insufficient guidance for accurate 3D face reconstruction.

Despite the initial concept's promising potential, the solution requires additional refinement. Future work should consider incorporating more comprehensive loss functions, such as photometric or identity-based losses, as well as including texture data in the generated faces.

6.2 Chapter 5 Conclusion

To test the original hypothesis—that increasing the diversity of training data improves the accuracy of 3D face recognition models—three experiments were conducted. The results clearly show that less data means less data diversity and thus, lower accuracy.

In the first experiment, the model achieved 100% accuracy using neutral facial expressions alone. However, it's crucial to recognize the limitation of such results in real-world scenarios. Facial expressions encountered in daily life are inherently diverse, including a broad spectrum of poses beyond neutral.

Therefore, future research should aim to enhance the model's robustness by training with a wider range of expressions and poses. Moreover, including texture data and improving loss functions will significantly contribute to creating a more realistic and accurate 3D face recognition model.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015. Software available from tensorflow.org. Available at: <https://www.tensorflow.org/>.

[2] Blanz, V. and Vetter, T. *A Morphable Model For The Synthesis Of 3D Faces* [online]. Max Planck Institute for Intelligent Systems, august 1999 [cit. 2023-24-12]. Available at: <https://cseweb.ucsd.edu/~ravir/6998/papers/p187-blanz.pdf>.

[3] Egger, B., Smith, W. A. P., Tewari, A., Wuhrer, S., Zollhoefer, M. et al. 3D Morphable Face Models—Past, Present, and Future. *ACM Trans. Graph.* New York, NY, USA: Association for Computing Machinery. jun 2020, vol. 39, no. 5. DOI: 10.1145/3395208. ISSN 0730-0301. Available at: <https://doi.org/10.1145/3395208>.

[4] Feng, Y., Feng, H., Black, M. J. and Bolkart, T. Learning an Animatable Detailed 3D Face Model from In-the-Wild Images. *ACM Transactions on Graphics (ToG), Proc. SIGGRAPH.* august 2021, vol. 40, no. 4, p. 88:1–88:13.

[5] Geng, J. Structured-light 3D surface imaging: a tutorial. *Adv. Opt. Photon.* Optica Publishing Group. Jun 2011, vol. 3, no. 2, p. 128–160. DOI: 10.1364/AOP.3.000128. Available at: <https://opg.optica.org/aop/abstract.cfm?URI=aop-3-2-128>.

[6] Genova, K., Cole, F., Maschinot, A., Sarna, A., Vlasic, D. et al. Unsupervised Training for 3D Morphable Model Regression. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2018.

[7] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition.* 2015.

[8] Henderson, P. and Ferrari, V. Learning Single-Image 3D Reconstruction by Generative Modelling of Shape, Pose and Shading. *International Journal of Computer Vision.* 2019. DOI: 10.1007/s11263-019-01219-8. Available at: <https://doi.org/10.1007/s11263-019-01219-8>.

[9] Ichim, A.-E., Kadlecek, P., Kavan, L. and Pauly, M. Phace: Physics-based Face Modeling and Animation. *ACM Trans. Graph.* 2017, vol. 36, no. 4.

[10] Jiang, C., Lin, S., Chen, W., Liu, F. and Shen, L. PointFace: Point Set Based Feature Learning for 3D Face Recognition. In: *2021 IEEE International Joint Conference on Biometrics (IJCB).* 2021, p. 1–8. DOI: 10.1109/IJCB52358.2021.9484368.

[11] Jing, Y., Lu, X. and Gao, S. 3D face recognition: A comprehensive survey in 2022. *Computational Visual Media*. december 2023, vol. 9, no. 4, p. 657–685. DOI: 10.1007/s41095-022-0317-1. ISSN 2096-0662. Available at: <https://doi.org/10.1007/s41095-022-0317-1>.

[12] Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F. et al. Practice and Theory of Blendshape Facial Models. In: Lefebvre, S. and Spagnuolo, M., ed. *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. DOI: 10.2312/egst.20141042. ISSN 1017-4656.

[13] Li, T., Bolkart, T., Black, M. J., Li, H. and Romero, J. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*. 2017, vol. 36, no. 6, p. 194:1–194:17. Available at: <https://doi.org/10.1145/3130800.3130813>.

[14] Liu, Y., Shi, H., Shen, H., Si, Y., Wang, X. et al. A New Dataset and Boundary-Attention Semantic Segmentation for Face Parsing. In: *AAAI Conference on Artificial Intelligence*. 2020. Available at: <https://api.semanticscholar.org/CorpusID:214302989>.

[15] Loper, M., Mahmood, N., Romero, J., Pons Moll, G. and Black, M. J. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*. ACM. october 2015, vol. 34, no. 6, p. 248:1–248:16.

[16] Ma, D. S., Correll, J. and Wittenbrink, B. The Chicago face database: A free stimulus set of faces and norming data. *Behavior Research Methods*. Dec 2015, vol. 47, no. 4, p. 1122–1135. DOI: 10.3758/s13428-014-0532-5. ISSN 1554-3528. Available at: <https://doi.org/10.3758/s13428-014-0532-5>.

[17] Ma, D. S., Kantner, J. and Wittenbrink, B. Chicago Face Database: Multiracial expansion. *Behavior Research Methods*. Jun 2021, vol. 53, no. 3, p. 1289–1300. DOI: 10.3758/s13428-020-01482-5. ISSN 1554-3528. Available at: <https://doi.org/10.3758/s13428-020-01482-5>.

[18] Ma, D. S., Correll, J. and Wittenbrink, B. The Chicago Face Database: A free stimulus set of faces and norming data. *Behavior Research Methods*. 2015, vol. 47, no. 4, p. 1122–1135.

[19] Oztireli, C., Valentin, J., Keskin, C., Pidlypenskyi, P., Makadia, A. et al. TensorFlow Graphics: Computer Graphics Meets Deep Learning. In:. 2019.

[20] Qi, C. R., Yi, L., Su, H. and Guibas, L. J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *ArXiv preprint arXiv:1706.02413*. 2017.

[21] Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H. A. A. K. et al. *PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies*. 2022.

[22] Sanyal, S., Bolkart, T., Feng, H. and Black, M. Learning to Regress 3D Face Shape and Expression from an Image without 3D Supervision. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. June 2019, p. 7763–7772.

[23] Szabó, A., Meishvili, G. and Favaro, P. Unsupervised Generative 3D Shape Learning from Natural Images. *ArXiv preprint arXiv:1910.00287*. 2019.

[24] Yang, S.-P., Seo, Y.-H., Kim, J.-B., Kim, H. H. and Jeong, K.-H. Optical MEMS devices for compact 3D surface imaging cameras. *Micro and Nano Systems Letters*. december 2019, vol. 7. DOI: 10.1186/s40486-019-0087-4.

[25] Zhang, Z., Da, F. and Yu, Y. Learning directly from synthetic point clouds for "in-the-wild" 3D face recognition. *Pattern Recognition*. 2022, vol. 123, p. 108394. DOI: https://doi.org/10.1016/j.patcog.2021.108394. ISSN 0031-3203. Available at: <https://www.sciencedirect.com/science/article/pii/S0031320321005732>.