Transforming Facial Landmarks for Virtual Avatar Facial Animation

Ole Algoritme

Thesis submitted for the degree of
Master in Programming and System Architecture
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Transforming Facial Landmarks for Virtual Avatar Facial Animation

Ole Algoritme

Abstract

Training police officers to conduct interviews with child abuse victims is an intricate, expensive, and time-consuming endeavor. We recognize the possibility of using 3D virtual avatars and facial mimicry in training programs that can assist law enforcement personnel with interviewing abused children. Transferring expressions and intricate nuances found in human faces onto a 3D virtual avatar, is one of the most complex tasks within computer vision. Despite significant progress and recent advancements in this field, we are still a long way from achieving virtual humans that are indistinguishable from real humans. An important aspect of creating realistic looking 3D virtual avatars is capturing the essence of complex human facial expressions. However, this requires expensive equipment that can capture and transfer facial mimicry from video onto a 3D virtual avatar.

In this thesis, we focus on the research and development of tools that utilizes algorithms and methodologies for facial expression capture and subsequent transfer onto 3D virtual avatars. We present evaluations of various animation techniques that employ facial landmark captures derived from video recordings, live video feeds, and images. Additionally, we showcase our research and implementation of 3D virtual avatar animation tools. Furthermore, we assess the effectiveness of utilizing distance algorithms for computing blendshape weights derived from facial landmarks by quantifying similarity between facial expressions. Our findings holds the potential to automate blendshape weight annotations of facial images. Lastly, we develop a dataset that includes face point clouds and sculpted 3D blendshape face models, which can be utilized in machine learning pipelines for blendshape weight predictions or facial analysis research.

Acknowledgments

This thesis was completed with the encouragement and help of supervisors Steven Hicks, Michael Riegler, and Pål Halvorsen.
I would like to express my deepest appreciation to you, for your guidance and support during the entire course of this thesis.

Contents

Glossary

CNN CNN (short for Convolutional Neural Network) is a type of deep learning algorithm used in machine learning for image and video analysis. CNNs are designed to recognize patterns in images by processing them through a series of convolutional layers and pooling layers.. 13, 50

CPU CPU (short for Central Processing Unit) is a central piece of electronics hardware that executes instructions from programs or operating systems.. 22

FACS FACS (short for Facial Action Coding Systems), is a system that categorizes human facial expressions into sets of shapes that can be described by Action Units.. 15

FFHQ FFHQ (short for Flickr-Faces-HQ) is a high-quality human faces image dataset. Originally created as a benchmark for generative adversarial networks (GAN), by NVIDIA Corporation. The FFHQ license is made available under Creative Commons BY-NC-SA 4.0 license.. 4

FPS FPS (short for Frames Per Second) is term that describes the frequency at which consecutive images or frames are captured or displayed.. 22

GPU GPU (short for Graphics Processing Unit) is a piece of computer hardware originally designed to accelerate graphics rendering, but now also widely used within machine learning tasks.. 22

inference Inference within the realm of Artificial Intelligence is the process of utilizing a neural network to yield results.. 22

LiDAR LiDAR (short for Light Detection and Ranging) is a remote sensing method that uses light in the form of a pulsed laser to measure variable distances to the Earth. The data collected using LiDAR can be environmental 3D objects, which are processed in machine learning pipelines.. 17

NLP NLP (short for Natural Language Processing) is a the ability given to a computer to understand text the same way as humans do.. 10

RAM RAM (short for Random Access Memory) is the fastest memory in a computer, which temporarily holds data that it needs to store and retrieve quickly.. 22

SVM SVM (short for Support Vector Machines) is a supervised machine learning algorithm, meaning they require labeled data to train on. The goal of SVMs is to find the hyperplane that maximizes the margin between two classes in a dataset.. 13

Chapter 1

Introduction

1.1 Background

We see the need for a long-term and result-oriented training program that gives law enforcement personnel the education and skill-set they need to be able to perform well-structured interviews with children suspected of being subjected to sexual abuse. Studies show that both basic and extensive forensic interview training that police officers undergo already incorporate interview training which teaches the most effective way of approach to get the most accurate narrative and the most amount of information [12]. The benefits of using open-ended questions over option and/or suggestions-based questions are generally well-known by individual officers. However, the same studies also show that although the interviewer is trained in the efficiency of asking open-ended questions, especially in interviews with children, the method is still not practiced in the actual interviews. In these emotional settings, the interviewer tends to base the conversation on giving the child options and suggestions of how a situation might have happened. What we can understand from the study is that it is essential to improve the way very young abuse victims are communicated with, and that it is a necessity to create an interview training program that is capable of showing trainees the differences in quality and quantity of information depending on how questions are asked.

The project aims to result in improved emotional and intellectual preparation for investigators to decrease the use of undesirable questioning techniques, and to increase the use of open-ended questioning that invites the child to freely tell their story and share their memories of the suspected abuse. There needs to be a better understanding of how to maximize the amount of valuable information received from the child [33].

1.2 Motivation

The training program aims to consist of a life-like 3D virtual avatar in the form of a child that is capable of independently interacting with the police trainee in an interview situation. To achieve this, the avatar needs to combine an array of technologies that can generate both customizable responses as well as realistic facial expressions. The training program is currently based on different chatbot technologies that are used both for material input, machine learning and structuring of the output [46]. For this program to be successful, it is important for the training to contain a wide variety of generated interactions. The other major challenge is to train investigators to be able to pick up on the emotional signs of abuse that can be conveyed through facial expressions and other non-verbal communication.

There needs to be better practice for suggestive questioning during child interviews, and there needs to be a better understanding of how to maximize the amount of correct information received from the child [44]. We are looking to improve the way police are trained by researching some of the most complex tasks in computer vision, specifically facial geometry and driving of realistic facial movements of an avatar. This thesis will focus on experimentation and development using different approaches and available technologies to create a realistic virtual human avatar. Our objective is to gradually increase the complexity as we see the results from our experiments and get closer to what we can define as realistic.

The motivation behind this thesis is to decrease the number of overturned convictions and failed prosecutions that result from unfortunate police interview practices of sexually abused children, as previously seen in highly publicized cases [65]. Abused children suffer in silence. Children are traumatized to the extent of central nervous system impairment, life-long anxiety and depression - and even suicides [34], due to sexual and physical abuse.

1.3 Problem Statement

Based on the motivation described in the previous section, we see the need for a training program that gives law enforcement personnel the ability to improve their skill set for conducting interviews with abused children. Creating an interactive avatar with realistic facial movements for use in the training program is a hard task. The process of transferring human facial mimicry is seen in Hollywood movies like Avatar, where a hired actor is equipped head to toe with a million-dollar suit consisting of cameras and sensors to detect face and body movements. Technologies already exist within the field of facial capture to avatar animation, but these solutions are proprietary and closed source, like Apple's ARKit [53]. ARKit's other limitation lies in requiring a depth sensor, rendering it unsuitable for detecting facial expressions and movements in previously recorded child interview videos that lack depth information. Other alternatives apply machine learning for driving facial expressions, relying on synthesizing of faces. Studies in this field have shown that synthesizing a child avatar has given imprecise results, ranging from blending noise as seen in [64] and inaccurate synchronizing of mouth shapes seen in [7].

The main problem we are trying to solve is the capture and transfer of facial mimicry from pre-recorded videos, live video feeds and facial images, using inexpensive cameras without depth sensors to drive a 3D virtual avatar inside the police interview training program. To solve this, we derive our four research objectives in this thesis;

Objective 1: *Research and develop a proof-of-concept implementation for facial expression transfer from video using facial landmarks.*

Provide a proof-of-concept implementation that extracts facial landmarks from pre-recorded videos of humans and apply them to a 3D face model. This approach can be utilized in the police training program by mapping the facial expressions and movements of an abused child, as observed in police interview videos, onto a 3D face model.

Objective 2: *Research and develop a proof-of-concept implementation for animating 3D virtual avatars using blendshape weights.*

Develop a tool that calculates blendshape weights using Euclidean and Euler calculations on detected facial landmarks from a web camera input stream, and transfer these onto a 3D virtual avatar for animation. This method holds the potential to animate a 3D virtual avatar in real-time from an inexpensive web camera, making it possible for hired actors to remotely control a 3D virtual avatar inside the police training program.

Objective 3: *Develop a dataset of facial landmarks and 3D blendshape face models for use in machine learning pipelines, and experiment with machine learning architectures utilizing such dataset.*

Research and develop a dataset consisting of facial landmark captures and 3D blendshape face models, to be used in facial expression analysis and machine learning pipelines for blendshape weight prediction. Experiment with machine learning architectures that utilize the created dataset.

In order for blendshape weight prediction to yield accurate results on facial images, a large quantity of data is imperative to construct a dependable and precise machine learning model.

Objective 4: *Research and develop a tool for automatic annotation of blendshape weights from facial landmark captures in facial images.*

Research and develop a tool that automatically annotates facial landmark captures in facial images with blendshape weights. Evaluate effectiveness of distance algorithms by creating a visualization tool that renders a 3D virtual avatar with the blendshape weight results.

1.4 Scope and Limitations

Based on the problem statement and its objectives, the scope of this thesis is research and development toward 3D virtual avatar animation, without using depth sensors or expensive equipment. We start by capturing facial landmarks using the open-source machine learning framework MediaPipe. The initial research starts by transferring facial landmarks found in pre-recorded videos onto a 3D face model. We extend the knowledge gained in the initial research to use Euclidean and Euler calculations to calculate blendshape weights, and use these to implement animation in a 3D virtual avatar with blendshape support. Furthermore, we develop a dataset for use in machine learning pipelines. The dataset consists of facial landmark captures done with FFHQ [32] accompanied by a set of 3D blendshape models. We experiment with different implementations of machine learning architectures using our newly created dataset. Lastly, we evaluate the effectiveness of different distance algorithms used in point clouds, for automatic blendshape weight annotation using facial images.

1.5 Research Methods

In general terms, a research method is a systematic approach used to collect, analyze or experiment with data for answering research questions or testing hypotheses to reach a research objective. Research methods can be qualitative, quantitative, or a mix of both, depending on the data and analyzation methods used.

Our research methodology is based on the Association for Computing Machinery (ACM) method. In 1989, a task force was assigned by the ACM Education Board to compile the core and fundamentals of computer science, which resulted in the report named "Computing as a Discipline" [25]. The report proposed a new framework for understanding computing as a discipline, and how it's being split into three paradigms that are essential components of computing; *theory*, *abstraction* and *design*.

The work presented in this thesis touches upon these paradigms, work which is conducted with experimental and iterative prototyping. In the subsequent subsections we describe each of the aforementioned ACM paradigms from the report and how our work fits into it.

1.5.1 Theory

The first paradigm is rooted in mathematics and consists of four steps followed in the development of a coherent, valid theory, according to the report. The Researchers are expected to iterate these steps when errors, inconsistencies or unwanted results are discovered.

1. Characterize objects of study (definition).
2. Hypothesize possible relationships among them (theorem).
3. Determine whether the relationships are true (proof).
4. Interpret results.

1.5.2 Abstraction

The second paradigm is rooted in the experimental scientific method and consists of four steps that are followed in the investigation of a phenomenon, according to the report. Researchers are expected to iterate these steps when predictions disagree with experimental evidence.

1. Form a hypothesis.
2. Construct a model and make a prediction.

3. Design an experiment and collect data.
4. Analyze results.

The abstraction paradigm is supported by the discoveries made when transferring facial landmarks onto a 3D face mesh. Even though successful facial landmark transfers were accomplished, we deduced that the occurrence of jitter and instability in the landmarks gave unrealistic, unwanted results. These unwanted results subsequently produced iterative experiments, from calculating simple blendshape weights using Euclidean/Euler calculations to point cloud distance measurements performed on the landmarks detected in facial images.

1.5.3 Design

The third paradigm is rooted in engineering and consists of four steps followed in the construction of a system (or device) to solve a given problem, according to the report. Researchers are expects to iterate these steps when tests reveal that the latest version of the system does not satisfactorily meet the requirements.

1. State requirements.
2. State specifications.
3. Design and implement the system.
4. Test the system.

The design paradigm is supported by a proof-of-concept system created to calculate blendshape weights from face point clouds, utilizing distance algorithms. The system is initially implemented and tested with a single blendshape. Subsequently, complexity is gradually increased by adding more blendshape weights and experimenting with different distance algorithms.

1.6 Ethical Considerations

When conducting research as a software engineer we need to take certain precautions and considerations into account. In general, we need to follow a set of principles that adhere to a certain code of conduct when collecting user data, releasing software that potentially could be misused, or the impact a software failure can introduce.

Our work focused on research in the field of facial expression transfer from human face to avatar, supporting police interview training of abuse children, which involves key considerations made when conducting research, namely:

- Protecting the privacy concerns of individuals seen in facial images or videos.
- Our work aimed at supporting the police interview training program, but could be misused.

- Children will suffer should the police interview training program fail.
- A very accurate solution for transfer of facial mimicry to virtual avatar is potentially dangerous in the hands of child predators.

In order to address privacy concerns, we made the decision not to utilize any pre-recorded video interviews of abused children. While our research serves as a starting point for developing a virtual avatar in the training program for police interviews, it is not a comprehensive solution. We believe that it was ethically appropriate to refrain from using these videos, as our research did not require them.

Regarding technological concerns, we recognize that the training program has the potential for misuse. The program aims to provide a software solution for training police officers in conducting interviews, but we acknowledge that it could be misused if it falls into the wrong hands.

Furthermore, our work reflects the technological considerations we took into account while developing our tools. A highly accurate virtual avatar that can be controlled using a simple web camera would be a desired software solution, but we also consider it to be a dangerous tool in the hands of child abusers.

We also consider the ethical importance of a comprehensive development and testing process for the training software, before being utilized in police interview training. Should such a software fail in any way, it could have a detrimental impact on affected children.

1.7 Main Contributions

Throughout this thesis, we conducted research and developed tools for transferring facial expressions from pre-recorded videos, live video feeds, and facial images. Our focus was on the development of a 3D virtual avatar that could be controlled by a hired actor through inexpensive camera equipment, or transfer facial mimicry captured from an abused child in pre-recorded interview videos.

This thesis aimed at creating a 3D virtual avatar for a training program to educate law enforcement personnel, enabling them to improve their skill-set and conduct well-structured interviews with children who are suspected of being subjected to sexual abuse. Our tools progressively increases complexity as we discover findings, drawbacks and limitations, within our experiments.

In this section we go through each of the research objectives we described in the problem statement stated in section 1.3 with a description of how we solved each of the stated objectives. Each objective is supported by experiments conducted using different algorithms and techniques mentioned throughout this thesis.

Objective 1: *Research and develop a proof-of-concept implementation for facial expression transfer from video using facial landmarks.*

The objective is supported by the development of a tool that detects facial landmarks in pre-recorded videos and subsequent transfer of facial landmarks onto a 3D face mesh. Using this tool we can transfer facial expressions from children in pre-recorded child abuse interviews onto a 3D face mesh.

Objective 2: *Research and develop a proof-of-concept implementation for animating 3D virtual avatars using blendshape weights.*

The objective is supported by the development of a tool that calculates the Eye-Aspect-Ratio (EAR) and Mouth-Aspect-Ratio (MAR) for measuring the opening and closing ratios of eyes and mouth, respectively. The ratios are utilized to compute blendshape weights, which are subsequently transferred to a 3D virtual avatar with blendshape support. For visualization support, we create a tool that integrates webcam input feed and displays blendshape names values corresponding to the calculated eyes and mouth opening and closing.

Objective 3: *Develop a dataset of facial landmarks and 3D blendshape face models for use in machine learning pipelines, and experiment with machine learning architectures utilizing such dataset.*

The development of PointFaces has supported the objective, which is a dataset consisting of facial landmark captures in various forms such as point cloud formats (PLY/PCD) and images (PNG). The dataset is accompanied by a set of sculpted 3D blendshape face models and can be utilized for further research in detecting or predicting various blendshapes from image or video inputs, as well as for facial expression analysis. PointFaces is made publicly available at Github, https://github.com/olealgoritme/pointfaces, and Zenodo, https://zenodo.org/record/7900081.

To further support the objective, we conducted experiments by implementing machine learning architectures such as Siamese One Shot and PointNet, which utilize the produced dataset.

Objective 4: *Research and develop a tool for automatic annotation of blendshape weights from facial landmark captures in facial images.*

To support our objective, we have created a proof-of-concept tool that leverages point cloud distance calculations to derive blendshape weights. We achieve this by utilizing K-D Tree Nearest Neighbor calculations on facial landmark captures obtained from our PointFaces dataset, which we developed specifically for predicting blendshape weights and analyzing facial expressions. By utilizing this tool, we can generate a 3D virtual avatar that displays the appropriate blendshape weights based on the input facial image. This work is made publicly available at Github, https://github.com/olealgoritme/master_thesis.

1.8 Thesis Outline

This thesis is organized into 7 chapters. The first two chapters provides the relevant introduction and background information, followed by four chapters that each are addressing one or more of the research objectives. The final chapter presents discussions, conclusions and future work.

The subsequent section provides an overview of each chapter and its contents.

Chapter 1: *Introduction*

This chapter provides the background and motivation for our thesis. It highlights our motivation to contribute to a police interview training program and discusses the lack of suitable interview techniques for abused children, as well as statistical data regarding the number of abused children. The chapter continues with presentation of the problem statement and its corresponding objectives.

Chapter 2: *Background on Facial Geometry, Facial Landmark Detection & Blendshapes for Virtual Avatar Animation*

The purpose of this chapter is to provide background information on the various topics that are relevant to the subsequent chapters. It covers the technologies and tools utilized in the experiments conducted throughout the thesis, with a particular emphasis on facial landmarks, facial geometry, blendshapes, and avatar animations.

Chapter 3: *Experiments on Facial Landmarks and Virtual Avatar Animation*

The chapter provides the initial experiments conducted with facial landmark detection and avatar animation. The first section introduces facial landmark captures from pre-recorded videos that are transferred to a 3D face model. The continuing section describes how we utilize different algorithms to derive blendshape weights by calculating the opening and closing of eyes and mouth. Lastly, we apply these derived blendshape weights to a 3D avatar model with blendshape animation support.

Chapter 4: *Dataset creation for Neural Networks: The PointFaces Dataset*

This chapter presents the work related to the creation of our dataset, which consists of sculpted blendshape 3D models and facial landmark captures in multiple formats. It covers a proposed pipeline on how the dataset can be utilized, and issues faced with scaling and transformation of face point clouds. The chapter continues to demonstrate alignment methods utilized in point cloud registration, as well as the blendshape 3D model sculpting process and encountered issues.

Chapter 5: *Experiments with Machine Learning for Point Clouds*

This chapter provides the experiments conducted with the implementation of two machine learning architectures. Both implementations utilize our dataset, which includes captures of facial landmarks and 3D models of blendshapes. The focus of this chapter is to assess the dataset's applicability and potential for predicting blendshape weights.

Chapter 6: *Implementations and Experiments with Point Cloud Distance Algorithms for Blendshape Weight Estimation*

This chapter presents the implementations and experimentation in utilizing different algorithms to measure distances between point clouds. We present the results obtained from distance computations performed on facial landmark captures and explain how we derive blendshape weights from them. The blendshape weights are applied to a 3D virtual avatar to evaluate similarity between the computed blendshape weight and the facial expressions seen in the input facial image.

Chapter 7: *Conclusion and Future Work*

This chapter presents conclusion of our thesis findings, as well as suggestions for improvements in future work.

Chapter 2

Background on Facial Geometry, Facial Landmark Detection & Blendshapes for Virtual Avatar Animation

This chapter focuses on the background information and related works for Facial Geometry, Facial Landmarks, Machine Learning, Blendshapes, Point Clouds, and Avatar Animations. We cover the technologies employed, their applications, and the utilized tools that are relevant to this thesis.

2.1 Machine Learning

Machine learning today is connected to what people think of as artificial intelligence. It is a large field within information technology, neurology, artificial intelligence, and other fields, where the end goal is to build a model that is a representation of large datasets. Customized algorithms are applied to datasets to allow computers to learn the desired outcome. Today's society has learned about machine learning through the appearance of a rather complex NLP model called ChatGPT [1], which seemingly changed the public opinion on what AI is and how it has the potential to automate complex tasks to make life easier. The following section describes the different machine learning types that are relevant for our thesis.

- **Supervised Learning** Supervised learning is identified by the usage of annotated training data, where a 'supervisor' provides guidance to the learning system regarding the labeling of training examples. The labeling typically consists of class labels in classification problems. Models are induced from these training data using supervised learning algorithms, which can then be applied to classify unlabeled data. The process of supervised learning involves creating a map between a set of input variables X and an output variable Y, which can be utilized to predict the outputs for new data. This technique holds a crucial position in machine learning and is of utmost significance in the processing of multimedia data [24].

- **Unsupervised Learning** Unsupervised learning is a type of machine learning in which the machine receives input data without any supervised target outputs or rewards from the environment. Despite the lack of feedback, unsupervised learning can be based on the idea of building representations of the input that can be useful for making decisions, predicting future inputs, and communication with other machines. Unsupervised learning is essentially concerned with identifying patterns in data that go beyond pure unstructured noise. Clustering and dimensional reduction are two simple examples of unsupervised learning [29].
- **Reinforcement Learning** Reinforcement learning is a type of machine learning that involves a machine interacting with it's environment through actions, which result in rewards or punishments. The aim is for the machine to learn to take actions that maximize the future rewards or minimize punishments over its lifespan. Reinforcement learning is closely related to decision theory and control theory, which deal with similar problems, and the solutions to these problems are often formally equivalent, although different aspects are emphasized [29].

Machine learning is related to our work through this thesis with the usage of facial landmark prediction models, our proposed machine learning pipelines, the dataset we create for machine learning purpose and experiments with machine learning architectural implementations.

2.2 Facial Geometry

Facial geometry refers to the study and measurement of the physical structure and features of the human face. It involves the analysis of the shape, size, and position of various facial elements such as the eyes, iris, nose, mouth, and jaw. Facial geometry plays a critical role in many applications, including biometrics, computer vision, and facial recognition technology. By analyzing the geometric features of a person's face, it is possible to create a unique identifier that can be used to identify an individual with a high degree of accuracy [47]. Facial geometry is also essential in the field of computer graphics, particularly in the creation of realistic 3D models of human faces. By accurately modeling the geometric structure of the face, it is possible to create lifelike animations and simulations that accurately represent the movements and expressions of the human face [62].

2.3 Facial Recognition

In the field of image analysis and computer vision, face recognition is a challenging and complex problem. It involves extracting features in the human face, regardless of lighting, expression, illumination, age, rotation, or pose [54]. Those features are analyzed to determine if the image contains a human face. As this paper focuses on extracting facial movements which can be further used to animate an avatar; face recognition is the fundamental and first stage of techniques used in the pipeline going from an input source image to extracted facial movements that are animated onto a virtual human.

2.4 Facial Landmarks

Facial Landmarks is the term referred to as the annotation drawn on an image with a human face, like seen in fig. 2.2. It shows the important areas of the face that are either manually annotated by researchers or developers, or automatically obtained through running an image through a machine learning model that has been trained to recognize facial geometry. The annotations are geometrical vertex points. Vertices are points where two or more edges, lines, or curves meet, as seen in fig. 2.1.

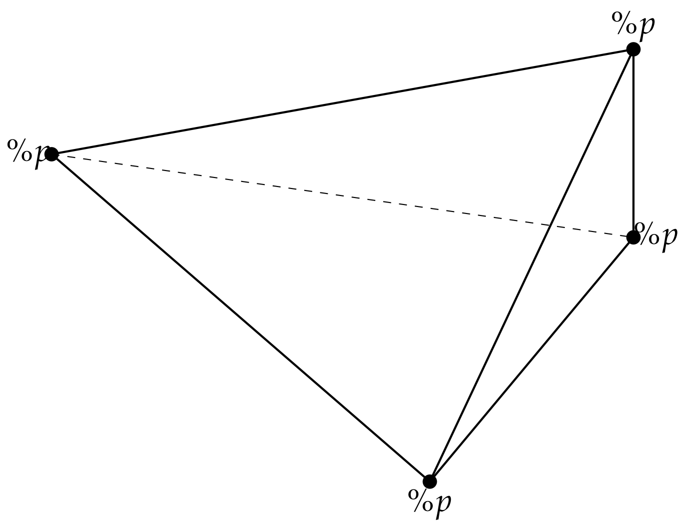


Figure 2.1: Vertices shown in 3D triangle. Each corner dot (A, B, C, D) represents a geometrical vertex point that connects to other vertices. Lines are drawn between these vertices to form a triangle.

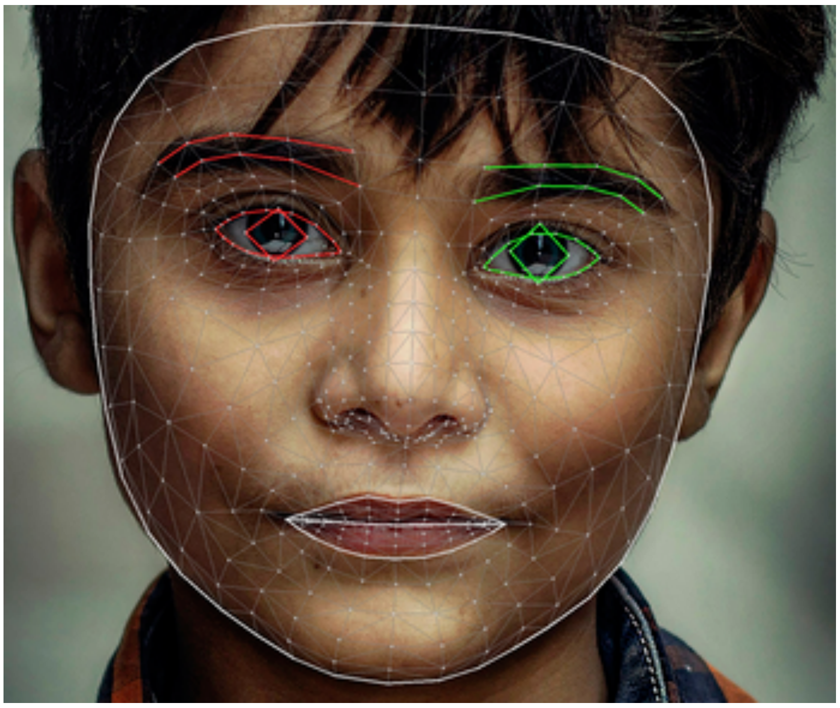


Figure 2.2: Facial landmarks applied to a static image. A total of 478 vertices connected by lines.

2.5 Facial Landmark Detection

Facial landmark detection is a computer vision technique that involves detecting and locating key points on a human face, such as the eyes, nose, and mouth. These key points are often referred to as landmarks and are used for various applications, including face recognition, facial expression analysis, and avatar animation. Facial landmark detection is typically accomplished using machine learning models, such as CNNs or SVMs, that are trained on large datasets of annotated facial images. Once trained, the models can be used to detect landmarks on new facial images. Facial landmark detection has numerous practical applications, such as in healthcare, where it can be used for diagnosing conditions such as Down syndrome and autism, and in augmented reality, where it can be used to overlay virtual objects onto a person's face [43, 55, 66].

Popular facial landmark detection frameworks that do not require depth sensors, are Dlib and MediaPipe. Both of these frameworks employ a machine learning model that detects facial landmarks, where Dlib detects 68 facial landmarks [8], MediaPipe detects 468 facial landmarks [35].

We utilize the MediaPipe framework for facial landmark detection throughout thesis, using it's Face Mesh module. The MediaPipe pipeline can be observed in fig. 2.3. The landmark detection model works by applying a UV visualization map of MediaPipe Face Mesh, depicted in fig. 2.4a to the detected face found in an image. Human faces are extracted from input sources, such as pre-recorded videos, static images or live video feeds such as web cameras. A face mesh is created using the runtime face metric landmarks as the geometry positions (XYZ), while both the texture coordinates (UV) and the triangular topology are inherited from the canonical 3D face mesh seen in fig. 2.4b.

To replicate facial movements from an input source human in 2D space onto a target 3D virtual avatar in 3D space, one must try to capture the different attributes of a human face. We are interested in testing MediaPipe for this purpose, as it has shown promise in annotating landmarks in real-time video feeds [69]. MediaPipe offers a cross-platform, customizable machine learning framework for live and streaming media [35]. The framework offers facial landmark detection through a neural network that is designed for real-time inference.

The MediaPipe model being used for detecting facial geometry is named Face Mesh. It is based on a pipeline with different steps in a machine learning model, as shown in the fig. 2.3. It captures the facial landmarks, which emphasize capturing landmarks around the eye and mouth area.

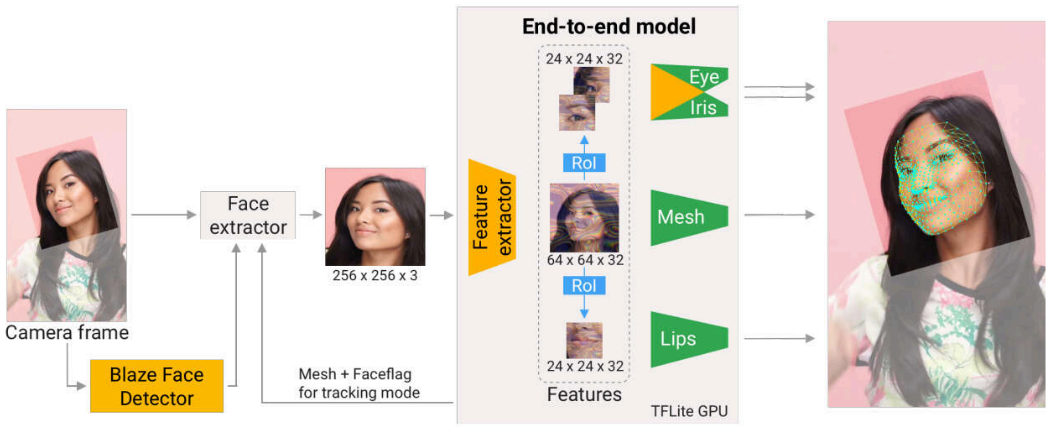
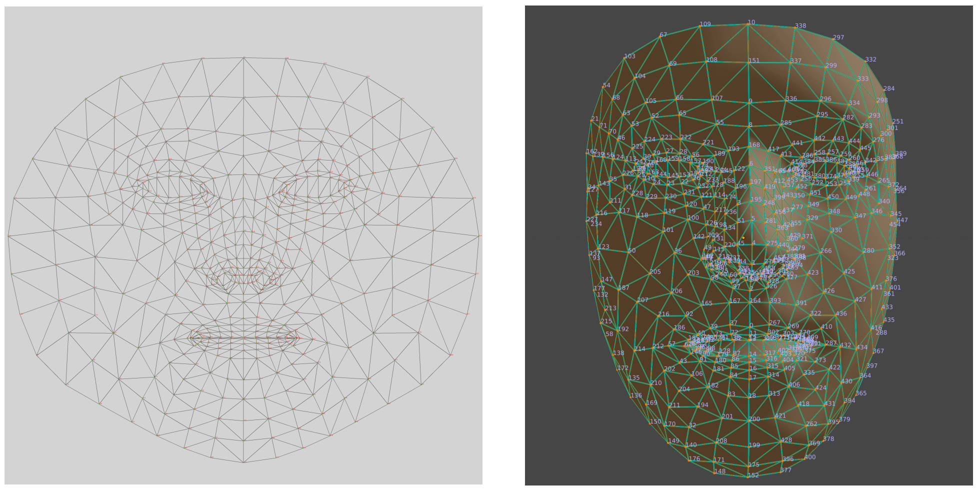


Figure 2.3: MediaPipe Face Mesh Pipeline. Figure taken from [35].



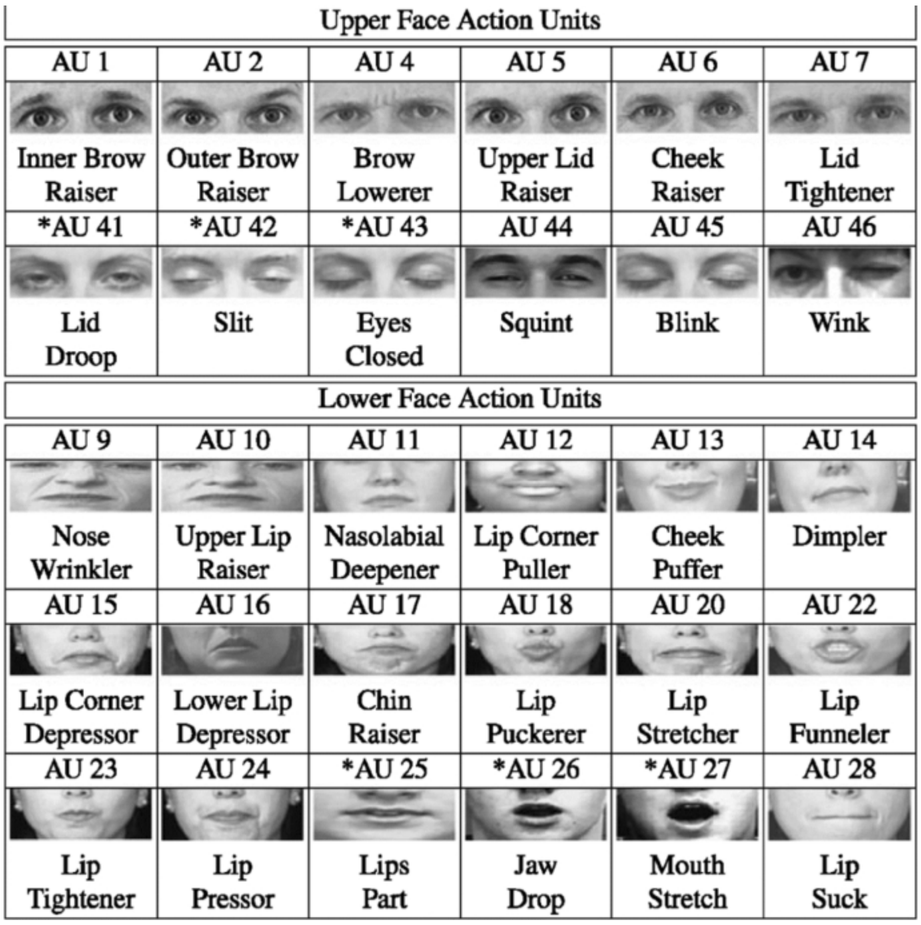
(a) UV Visualization map (b) 3D Face mesh model

Figure 2.4: Face Mesh Model. Figure taken from [30].

2.6 Blendshapes

A morph target or shape key is a technique for deforming a mesh from one shape to another, commonly known as a blendshape. The concept of blendshapes seen in fig. 2.5 is based on FACS, which categorizes human facial expressions into sets of shapes that can be described by Action Units. FACS was initially introduced by a Swedish anatomist [37], was further developed and released by Ekman/Friesen in 1978 [28], which was work derived from analysis of facial movements.

A blendshape is equivalent to an Action Unit in the FACS system. Blendshapes today are heavily used in the 3D industry, particularly to drive character facial animations. Each blendshape defines an interpolated animation between one shape and another, as illustrated in fig. 2.6.



Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Figure 2.5: Facial Action Units from the FACS system. Figure taken from [68].

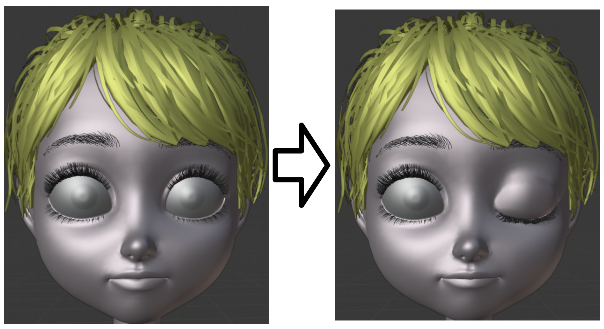


Figure 2.6: Deformation from neutral to blinking eye using blendshapes, a derivation from FACS Action Units.

2.7 Avatar Animation

Avatar animations refer to the creation of digital representations of individuals or characters, often used in various applications such as video games, movies, and virtual reality experiences. These animations involve the use of computer software to generate lifelike movements and actions of the avatars, enabling them to interact with their environments and other characters. The process of avatar animation typically involves designing the character's appearance, creating a digital skeleton or rig, and then animating the rig using a variety of techniques such as shape key animation, morph target animation or blendshape animation, all referring to the same technique of interpolating between a neutral and a fully expressed expression. This allows for the creation of realistic movements that mimic the way a human or animal would move in the real world.
Today, avatar animations are widely used in various industries, including entertainment, education, and healthcare. In video games, for example, avatar animations enable players to control characters and immerse themselves in virtual worlds. In education, avatars can be used to create engaging and interactive learning experiences, while in healthcare, they can be used to simulate medical procedures or train medical professionals. Avatar animations are a critical tool in modern computing, enabling developers to create compelling and realistic digital experiences that blur the line between the virtual and the real.

Avatar animations driven by facial landmark detection is getting more popular with today's focus on The Metaverse [22] and AR/VR/XR applications now that more powerful computer and mobile hardware is available at an average household disposal.

A study in this field is based on capturing facial expressions through a real-time 3D non-rigid tracking system [60]. Expression transfer is achieved, by combining a basic expression model with a synthetic approach that better capture person-specific characteristics. The drawbacks are that most of them exhibit only driving basic emotions; neutral, joy, anger, sad, surprise, fear and disgust, whereas the human face is far more complex than basic emotions.

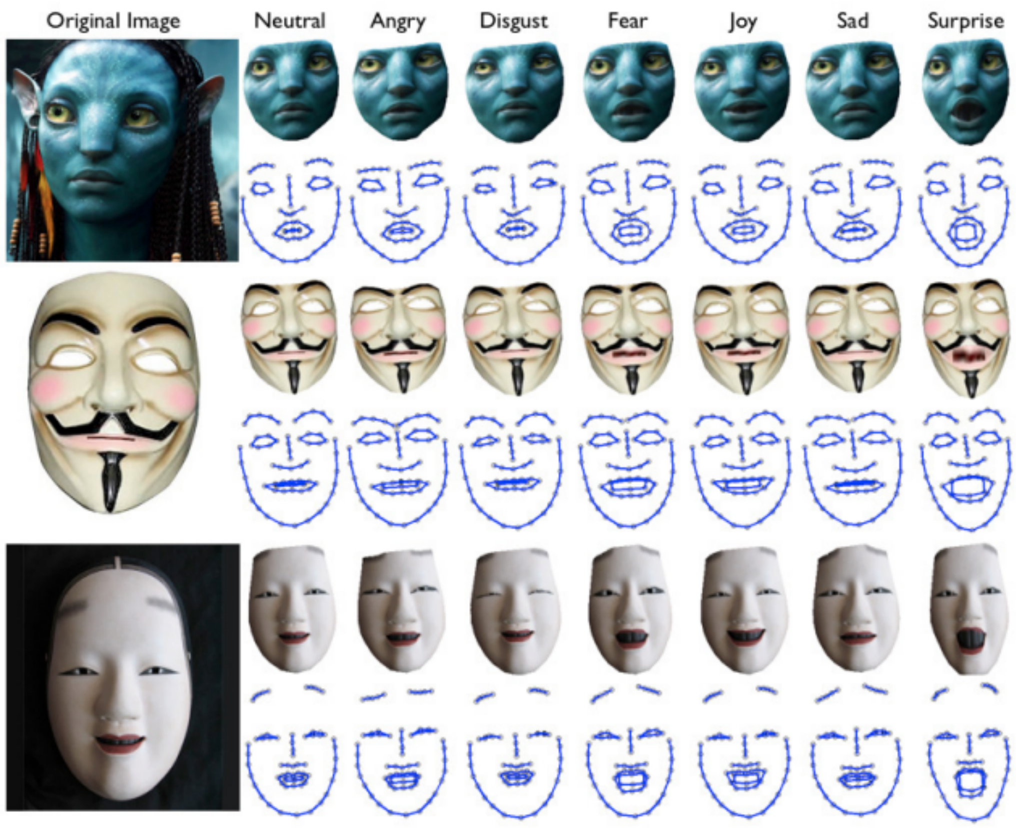


Figure 2.7: Semantic expression transfer. Figure taken from [60].

2.8 Point Clouds

A point cloud is a digital representation of a physical object, person, environment or anything that exists in the real world, which is created by capturing a large number of points in 3D space using various technologies such as LiDAR, photogrammetry, structured light scanning, cameras or similar [48].

Each point in a point cloud is defined by its geometrical coordinates and may contain additional data such as color, intensity and even light reflection. Point clouds are often used in fields such as architecture, engineering, and construction to create accurate 3D models of real-world objects or environments. They can also be used in robotics, computer vision, and virtual reality applications. Because point clouds are generated from real-world data, they are often noisy and may contain outliers or missing data. As a result, point cloud processing algorithms are often used to clean and organize the data, as well as to extract useful information such as surface geometry or object segmentation. Point clouds have many practical applications, such as creating digital twins of buildings or infrastructure, generating 3D maps of natural environments, or aiding in the navigation and control of autonomous vehicles [21]. For our use case, a point cloud represents points from facial landmark capture as seen in fig. 2.8. The inference done with MediaPipe gives us predictions in the form of a 3D mesh model that corresponds to the facial structure and expression from an input facial image in 2D, where the depth is predicted by the machine learning model, and the point cloud is the definition of all of the spatial coordinate points in that landmark facial capture.

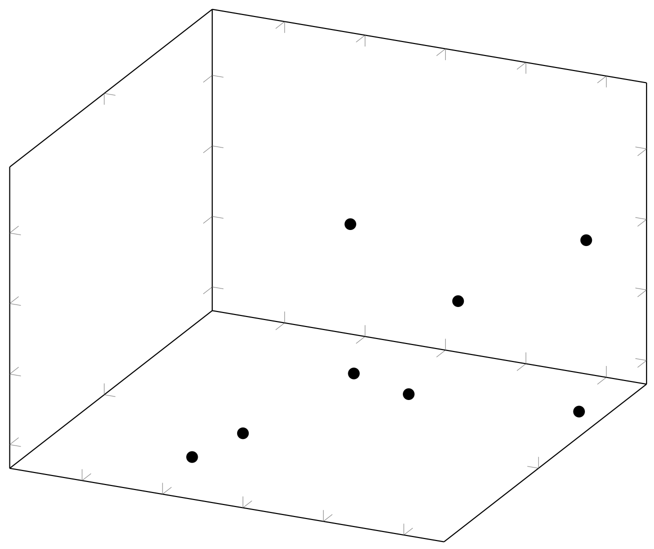


Figure 2.8: Three-dimensional Point Cloud with only geometry coordinates. No colors, intensity, normals or faces.

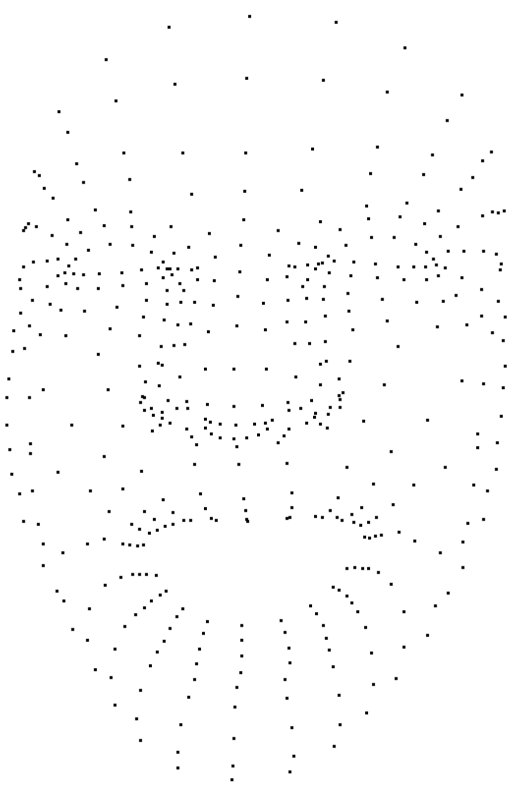


Figure 2.9: A blendshape seen as a Point Cloud. The Point Cloud contains geometry coordinates that are derived from a facial landmark capture.

2.9 Tools and technologies

This section represents the tools and technologies applied within this thesis.

2.9.1 Python

Most of the work done within this thesis relies heavily on the Python 3 programming language. Python 3 is the latest release of the Python programming language, released in 2008. It is a popular high-level programming language used in web apps, data analysis, artificial intelligence, machine learning and scientific computing. Python 3 offers new features and improvements over its predecessor, Python 2, including improved unicode support, enhanced syntax and a cleaner language design. According to the Python Software Foundation, the main goals of Python 3 are to improve the language's usability and eliminate redundancy and inconsistencies in the language's design [59] Most of the programming tasks in this thesis utilizes Python3.

2.9.2 PyTorch

PyTorch is a Python library that facilitates the development of deep learning/machine learning software. PyTorch is versatile and can be used for a wide range of applications, suitable for both novice deep learning programmers and professional real-world applications [38] PyTorch is used within this thesis in experiments with machine learning architectures.

2.9.3 Blender

Blender is a free and open-source 3D graphics software used for modeling, animation, rendering and visualization. It includes a real-time 3D viewer and a graphical user interface. Additionally, Blender provides a Python interface that enables users to load data into the program [41]. This thesis utilizes Blender for rendering, visualization and modeling of 3D face models.

2.9.4 Numpy

NumPy is a Python library used for numerical computing. It is designed to handle large datasets and perform mathematical operations efficiently. NumPy provides an N-dimensional array object, which is a fast and flexible data structure for storing and manipulating large arrays of numerical data. It also includes a range of functions for performing basic and advanced mathematical operations on arrays, such as linear algebra, Fourier transforms, and random number generation [63]. Our work utilizes NumPy for basic numerical compute, multidimensional arrays and in distance computing algorithms.

2.9.5 OpenCV

OpenCV is a computer vision library that provides tools for image and video processing, feature detection, machine learning, camera calibration, and GUI development. It is widely used in industries such as robotics, automotive, healthcare, and security [23]. During the course of this thesis, we utilized OpenCV for extraction of facial images from pre-recorded videos, live video feeds and image processing tasks.

2.9.6 Open3D

All point cloud related work within this thesis relies heavily on the Open3D library using its Python bindings. Open3D is an open-source library for 3D data processing, visualization, and machine learning. It provides a convenient and powerful interface for working with 3D data, including point clouds, meshes, and RGB-D images. Open3D is written in C++ and has bindings for Python, making it easy to use in a wide range of applications. Open3D offers a variety of algorithms and tools for 3D data processing, such as 3D registration, surface reconstruction, and point cloud segmentation. It also provides a flexible visualization framework for displaying 3D data and interacting with it in real-time. One of the main advantages of Open3D is its Python interface, which allows users to rapidly prototype and experiment with 3D data processing and machine learning algorithms. The Python API provides a high-level interface that abstracts away many of the low-level details of working with 3D data, allowing us to focus on research or application development [70]. Open3D is used extensively for all point cloud related work throughout this thesis.

2.10 Summary

In this chapter, we discussed the various fields, subject backgrounds, and related works that are pertinent to our four research objectives. Firstly, we introduced the concept of machine learning and its relevant categories. Additionally, we discussed the MediaPipe framework, which utilizes a machine learning model for facial landmark capture. The chapter also covered facial geometry, the working principle of facial landmark detection, and its primary application in avatar animations. We detailed what blendshapes are, and we highlighted the essential tools and technologies used in this thesis, including Open3D and Python 3. We also explained the significance of point clouds and avatar animations in contemporary society.

Chapter 3

Experiments on Facial Landmarks and Virtual Avatar Animation

This chapter focuses on the experiments conducted using facial landmark detection to drive animations of a 3D face, in accordance with both objective 1 and objective 2, as described in section 1.3. We employ MediaPipe, described in section 2.5, a machine learning framework for facial landmark detection, to obtain facial expressions and movements from an input source; like a video recording or a web camera feed. The obtained facial landmarks are utilized in two different ways in this chapter; the first section is devoted to applying detected facial landmarks to a 3D face mesh, and the second section is devoted to deriving blendshape weights from facial landmarks and applying these to a virtual avatar. We evaluate and summarize our findings in the chapter summary.

3.1 Facial Landmark Detection Performance Benchmark

To get a performance indication of the facial landmark detection with MediaPipe, we perform tests on a pre-recorded video, seen in fig. 3.1, to get reproducible results.

The tests are performed with three consecutive runs utilizing C++ and Python, with both GPU and CPU as devices for inference. The average is calculated and the performance metrics applied are FPS and inference time.

Tests are performed on a desktop computer consisting of the following components:

- CPU: AMD Ryzen 5950X (16 cores, 32 threads)
- GPU: RTX 3080 10GB GDDR6X (8704 Cuda cores, 272 Tensor cores)
- RAM: 64GB DDR4 NON-ECC

Observing the results, depicted in figs. 3.2 and 3.3, we clearly see that the GPU outperforms the CPU in both FPS and inference when running MediaPipe facial landmark capture on input video. We conclude that for realtime landmark capture there is a clear advantage to utilizing a GPU over a CPU.

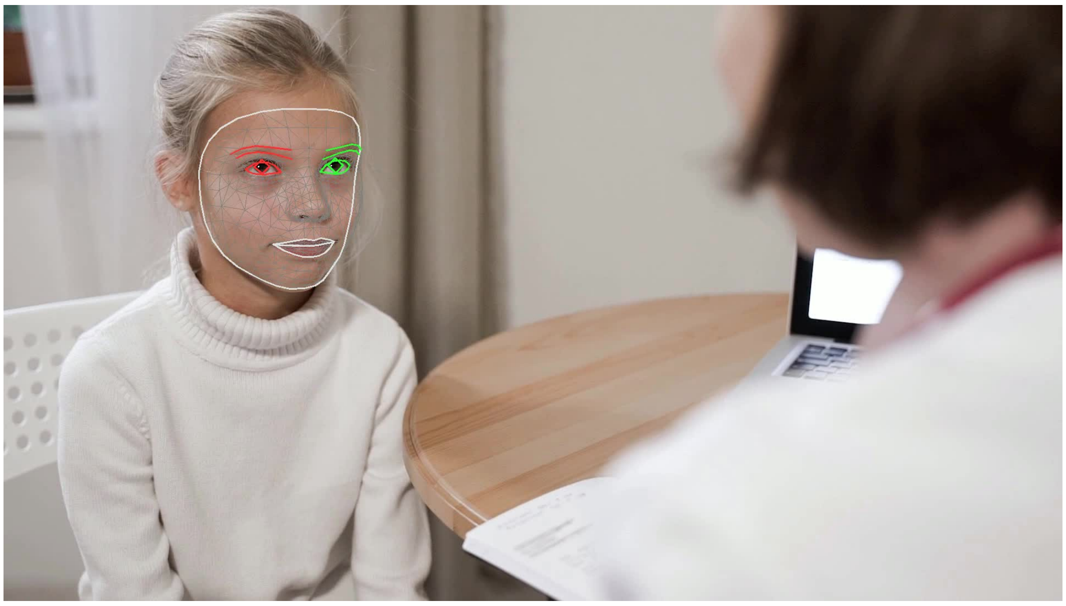


Figure 3.1: Facial landmarks detected from pre-recorded video. Source: [52] [1]

[1]Source Video: https://www.pexels.com/video/a-doctor-interviewing-the-patient-5998346 | License: Free to use

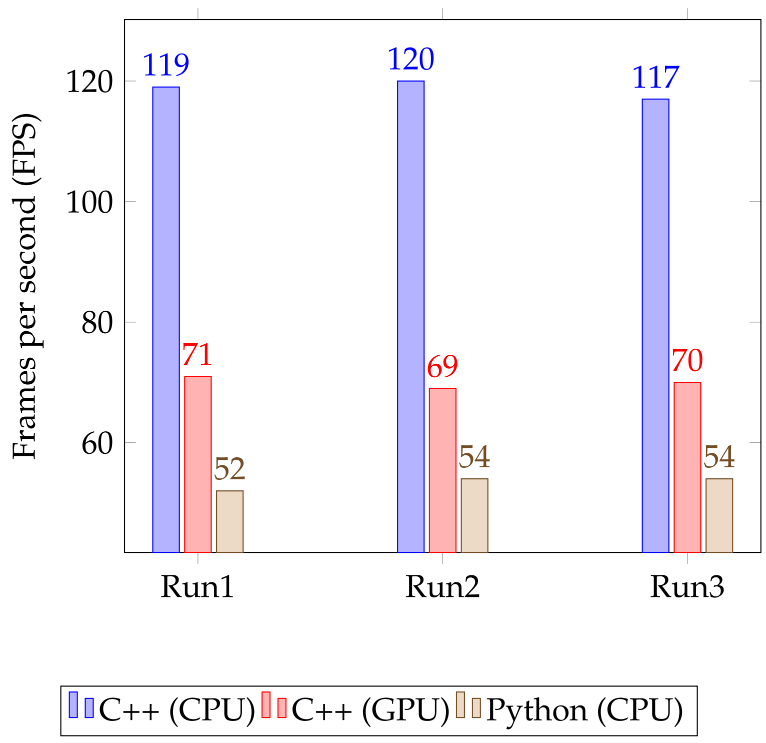


Figure 3.2: Results showing the number of frames per second from utilizing the MediaPipe Face Mesh machine learning model to predict facial landmarks on pre-recorded video shown in fig. 3.1.

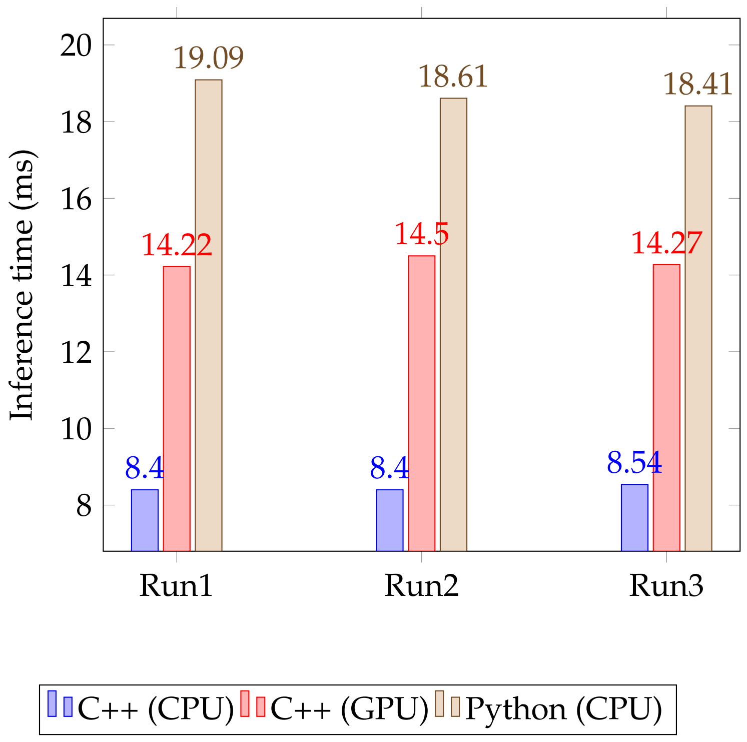


Figure 3.3: Inference time results from utilizing the MediaPipe Face Mesh machine learning model to predict facial landmarks on the pre-recorded video seen in fig. 3.1.

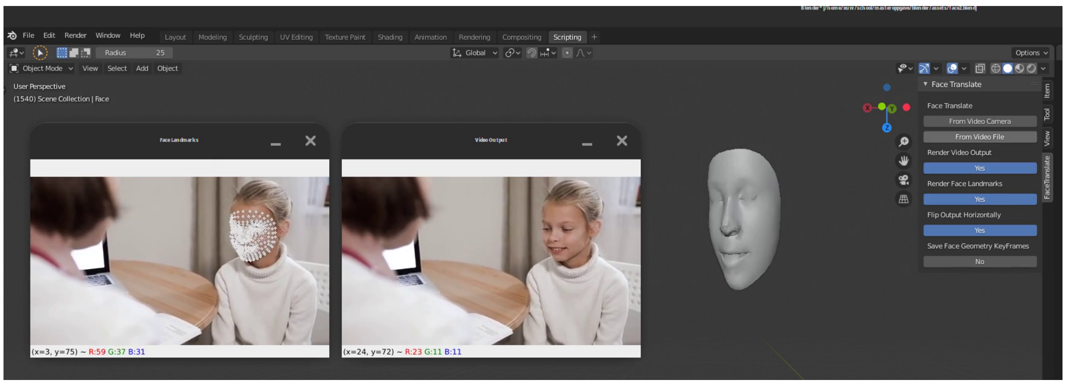
3.2 Applying Facial Landmarks to 3D Face Mesh

We developed a method for detecting facial landmarks from an input video and transferring them to a 3D face mesh, which has the potential to recreate a child's observed facial movements and expressions during interviews. To streamline this process, we created a Blender plugin that automates facial landmark detection and expression transfer, as depicted in figs. 3.4 and 3.5. This plugin was created using Python3, a popular high-level programming language, for rendering output using Blender, MediaPipe for facial landmark captures and OpenCV for image and video processing, described in sections 2.5, 2.9.1, 2.9.3 and 2.9.5, respectively.

To be able to visualize and evaluate the results of animating the 3D face mesh, our Blender plugin pipeline maps the corresponding outputs of facial landmark points to the points of a 3D face mesh. We employ facial landmark detection by running inference on the MediaPipe Face Mesh model, which outputs a vector of geometrical points corresponding to important areas of the face, as seen in fig. 2.3. The 3D face mesh from MediaPipe, seen in fig. 2.4b, is used as our 3D face mesh model for our animation purposes, chosen because this 3D model has the same vertex topology (geometrical points) that MediaPipe outputs.

The plugin pipeline starts with capturing facial landmarks from an input video using MediaPipe. Subsequently, facial landmark capture is processed frame-by-frame on the input video. Each extracted facial landmark point is individually matched to the 3D face mesh's topology, and for each point, the corresponding position in the 3D face mesh is updated in Blender's render view.

We observed that when utilizing our developed plugin, we could successfully transfer the facial expressions and movements from a person in a video input onto a 3D face mesh, as seen in the fig. 3.6. However, the predicted facial landmarks that MediaPipe outputs, are noisy and unstable, which is very difficult to portray in still images. There is a significant amount of jitter to each of the landmark outputs, which are not accurate to the source face and do not give realistic results. Attempts to mitigate these issues using the 1€ filter [11] were unsuccessful.



Face Translate

Face Translate

From Video Camera

From Video File

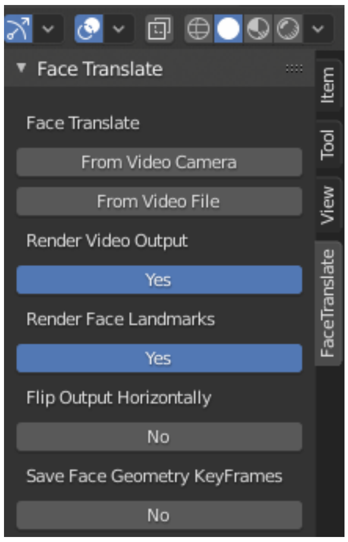
Render Video Output: Yes

Render Face Landmarks: Yes

Flip Output Horizontally: Yes

Save Face Geometry KeyFrames: No

Figure 3.4: Overview of Face Translation, the Blender plugin we created.


Face Translate

Face Translate

From Video Camera

From Video File

Render Video Output

Yes

Render Face Landmarks

Yes

Flip Output Horizontally

No

Save Face Geometry KeyFrames

No

Face Translation 3D UI VIEW UI controls to start Face Translation in Blender. Records facial landmarks from video camera or video file. Can also record facial landmarks to be replayed on the 3D face mesh and also view the render output from the Video in a separate window.

Figure 3.5: The UI View in Blender.

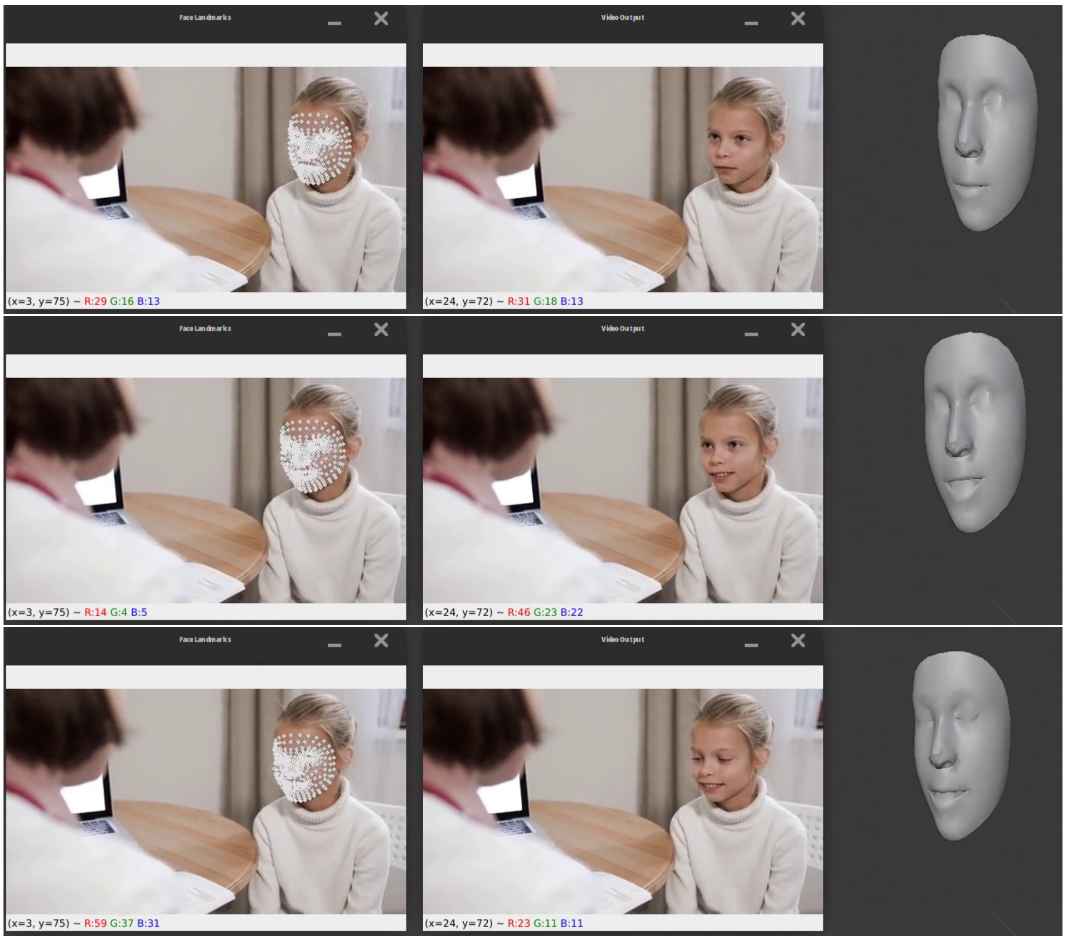

Face Landmarks
Video Output
(x=3, y=75) ~ R:29 G:16 B:13
(x=24, y=72) ~ R:31 G:18 B:13
Face Landmarks
Video Output
(x=3, y=75) ~ R:14 G:4 B:5
(x=24, y=72) ~ R:46 G:23 B:22
Face Landmarks
Video Output
(x=3, y=75) ~ R:59 G:37 B:31
(x=24, y=72) ~ R:23 G:11 B:11

Figure 3.6: Facial landmark transfer from input video to 3D Face mesh.

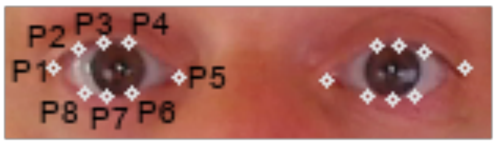
3.3 Virtual Avatar Animation using Blendshapes

Our experiments involve computing the open/close ratios of the eyes and mouth, which are then used to derive blendshape weights. As explained in section 2.6, blendshapes are a type of morph targets utilized in animation technology. The resulting blendshape weights can be transferred to 3D avatar models that support blendshape animation.

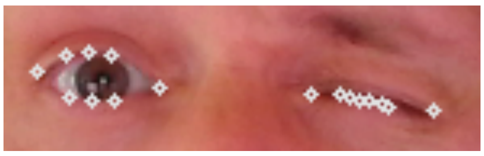
In section 3.2, our findings reveal that the direct mapping of the facial landmarks to 3D face mesh proved unstable and exhibited significant jitter. Consequently, we explore an alternative approach to transferring facial expressions and movements onto a virtual avatar, namely using blendshapes., In this section, we present the methods used to calculate blendshape weights from facial landmarks through simple calculations. To facilitate the process, we have developed a toolset that extracts blendshape weights from the eyes and mouth, visualizes the calculated blendshape weights, and applies them to a virtual avatar.

3.3.1 Eye Blink Calculation

For the transfer of eye blink/close animations onto a virtual avatar, we rely on Euclidean equations for calculating the distances between different points, as outlined in fig. 3.7. We use the points from the upper and lower eyelids as well as the corners of the eyes and calculate the distance ratio between them using the Eye Aspect Ratio (EAR) as proposed by the Czech Technical University[45]. Using the EAR calculation depicted in fig. 3.8, we estimate how much each eye is closed or open by finding the Gini coefficient, which constrains the value to be between 0 and 1 [26]. A value of 0 would indicate the eye is open, and a value of 1 would indicate the eye is closed, and 0.5 would indicate half way open.



(a) Eye Open



(b) Eye Closed

Figure 3.7: The eye points used in Eye Blink Calculation.

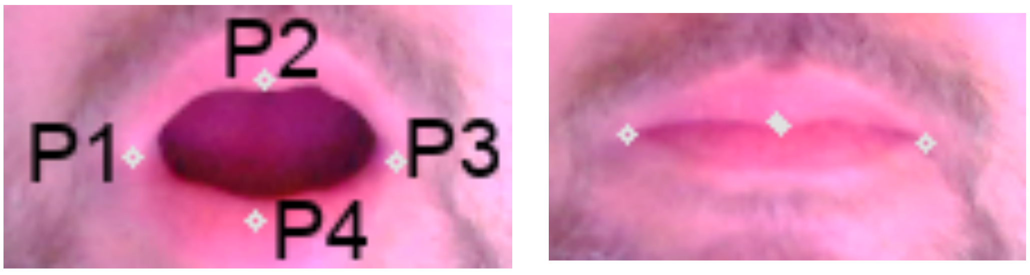
$$EAR = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|} \tag{3.1}$$

Figure 3.8: The EAR calculation. Where p_1 and p_5 denotes the corners of the eyes. p_2, p_3, p_4 the upper eye lids and p_6, p_7, p_8 for the bottom eye lids.

3.3.2 Mouth Open Calculation

Similar to the Eye Aspect Ratio, we employ a Euclidean equation to compute point distances and the corresponding distance ratios for the Mouth Aspect Ratio (MAR), seen in fig. 3.10. Additionally, we translate the mouth open indication to a range between 0 and 1, which indicates the extent to which the mouth is open.

The MAR algorithm is widely utilized in the automotive industry to identify driver drowsiness [61]. However, in our case, our focus is solely on determining the degree to which the mouth is open or closed, and therefore, we have excluded several of the detected points surrounding the mouth area, which are deemed unnecessary.



(a) Mouth Open (b) Mouth Closed

Figure 3.9: The mouth points used in Mouth Open Calculation.

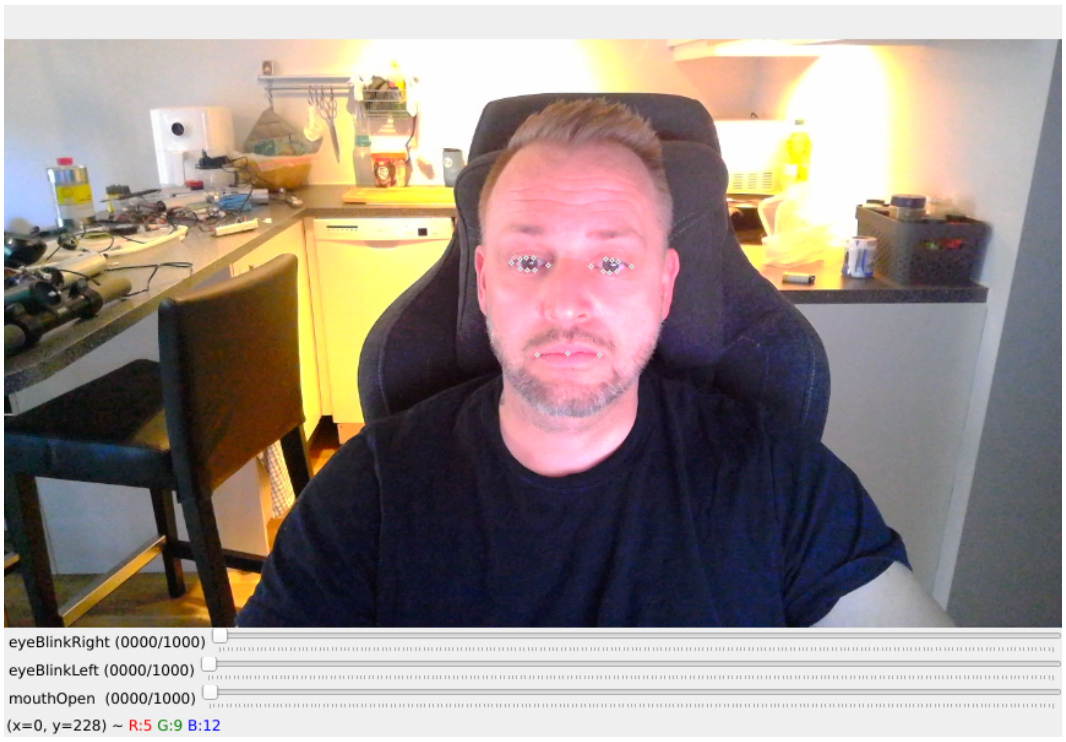
$$MAR = \frac{||p_2 - p_4||}{2\,||p_1 - p_3||} \tag{3.2}$$

Figure 3.10: The MAR formula. Where p_1 and p_3 denotes the corners of the mouth, p_2 upper lip, and p_4 lower lip.

3.3.3 Visualizing Blendshape Weights

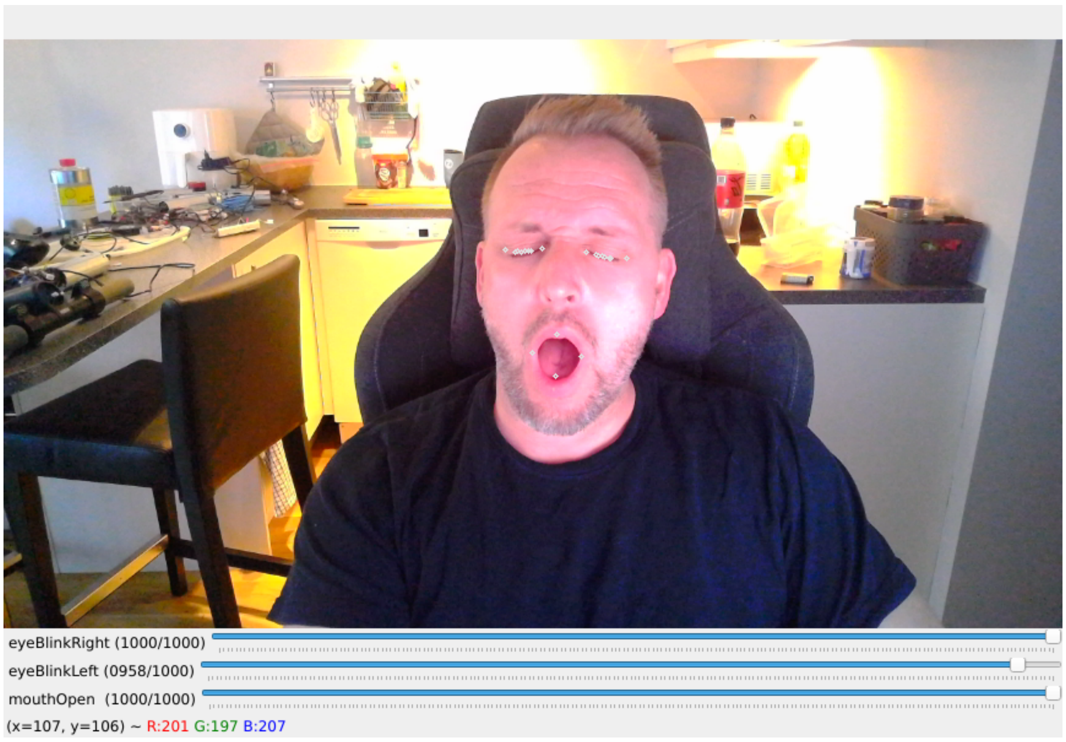
To facilitate the visualization of our EAR/MAR calculated blendshape weights, we developed a tool that utilizes trackbars to display the calculated weights on a scale of 0-1000. A weight of 0 indicates no detection of the corresponding blendshape, while a weight of 1000 represents full expression. The tool is designed to take input from a webcam, which enables easy and direct control of facial expressions.

We utilized the visualization tool to display the results of our EAR/MAR calculations for the eyes and mouth, as illustrated in figs. 3.11 to 3.14. The findings demonstrate our ability to successfully compute blendshape weights for eyeBlinkRight, eyeBlinkLeft, and mouthOpen, which can be applied to a 3D virtual avatar for animation purposes.



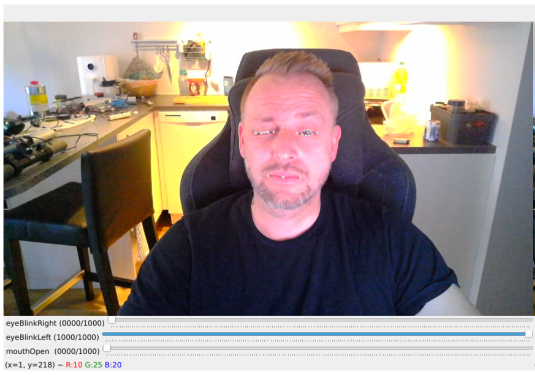
eyeBlinkRight (0000/1000)
eyeBlinkLeft (0000/1000)
mouthOpen (0000/1000)
(x=0, y=228) ~ R:5 G:9 B:12

Figure 3.11: Neutral Face, no detection

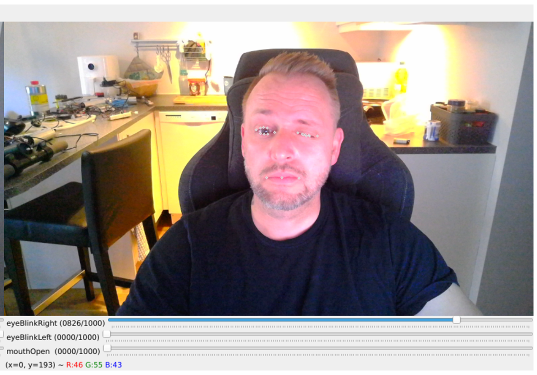


eyeBlinkRight (1000/1000)
eyeBlinkLeft (0958/1000)
mouthOpen (1000/1000)
(x=107, y=106) ~ R:201 G:197 B:207

Figure 3.12: Mouth Fully Open, both Left and Right Eyes Closed

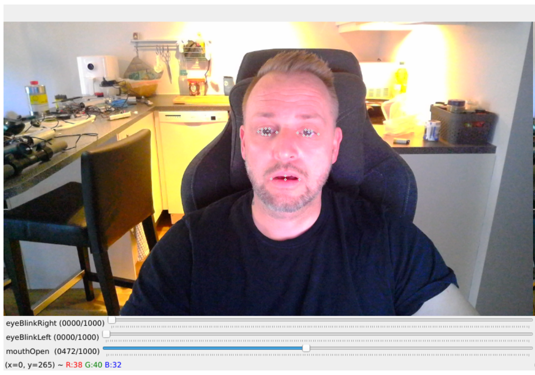

eyeBlinkRight (0000/1000)
eyeBlinkLeft (1000/1000)
mouthOpen (0000/1000)
(x=1, y=218) ~ R:10 G:25 B:20

(a) Eye Aspect Ratio - Left Eye Blink

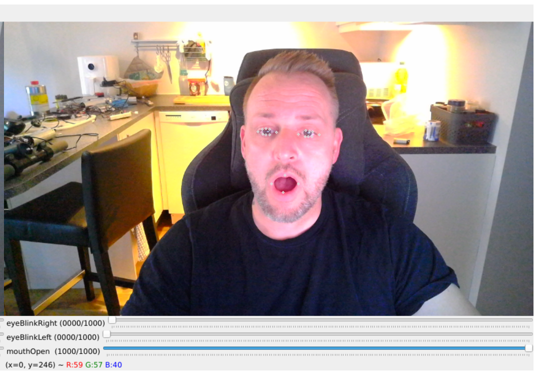

eyeBlinkRight (0826/1000)
eyeBlinkLeft (0000/1000)
mouthOpen (0000/1000)
(x=0, y=193) ~ R:46 G:55 B:43

(b) Eye Aspect Ratio - Right Eye Blink

Figure 3.13: Eye Aspect Ratio detection


eyeBlinkRight (0000/1000)
eyeBlinkLeft (0000/1000)
mouthOpen (0472/1000)
(x=0, y=265) ~ R:38 G:40 B:32

(a) Mouth Aspect Ratio - Halfway Open


eyeBlinkRight (0000/1000)
eyeBlinkLeft (0000/1000)
mouthOpen (1000/1000)
(x=0, y=246) ~ R:59 G:57 B:40

(b) Mouth Aspect Ratio - Fully Open

Figure 3.14: Mouth Aspect Ratio detection

3.3.4 Transferring Facial Movements to Virtual Avatar using Blendshapes

After being able to visualize our calculated blendshape weights, we experiment with transferring the calculated blendshape weights onto a 3D virtual avatar. This is done by translating the captured facial landmarks into blendshape weights using the respective EAR and MAR calculations. These blendshape deformation weights are applied to the respective blendshapes of the 3D virtual avatar model frame by frame, updating each point positions in the model mesh in each frame iteration.

The virtual avatar 3D model with blendshape support is a free sample model from PolyWink [56], seen in fig. 3.16.

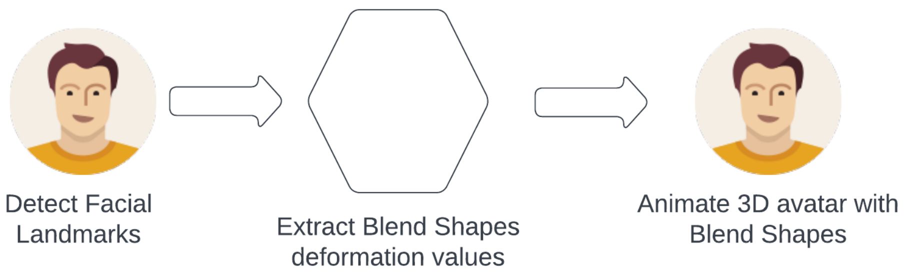

Detect Facial Landmarks
Extract Blend Shapes deformation values
Animate 3D avatar with Blend Shapes

Figure 3.15: Simplified overview of facial landmark detection to virtual avatar blendshape animation.

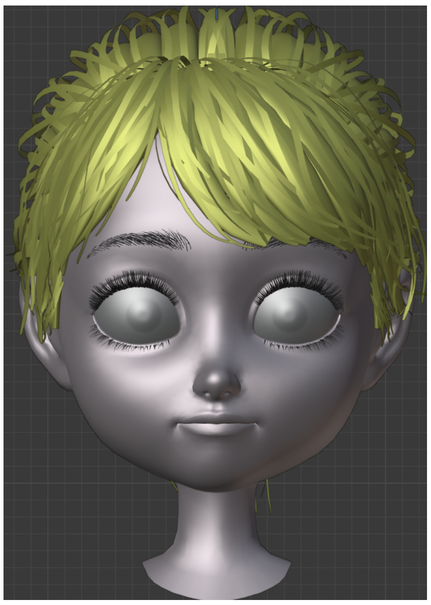
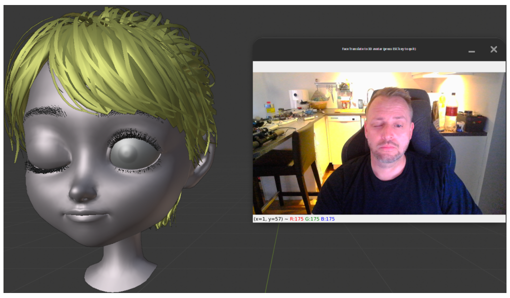
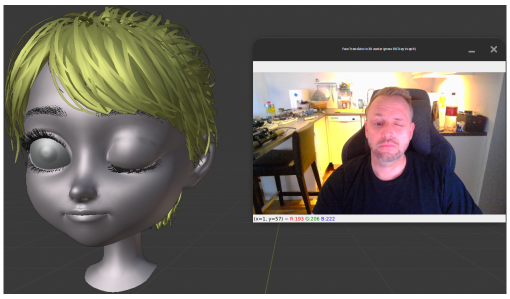


Figure 3.16: The virtual avatar 3D model with blendshape support. Source: [56]. [2]

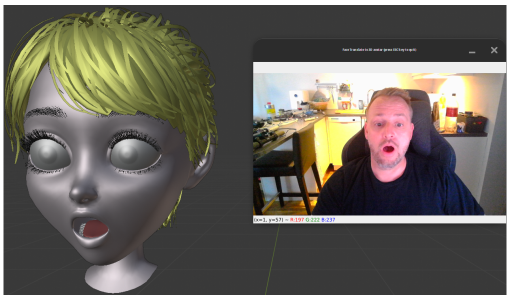
[2]Source Model: https://polywink.com/15-facial-animation-for-iphone-x.html | License: CC-BY-ND



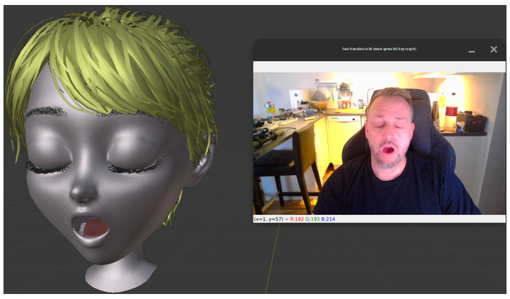
(a) Left Eye Blink.



(b) Right Eye Blink.



(c) Mouth Open.



(d) Mouth Open, Left/Right eye Closed.

Figure 3.17: Blendshape weights derived from our calculations, transferred to virtual avatar.

3.4 Summary

We carried out experiments to assess the effectiveness of facial landmark detection on pre-recorded videos and webcam feeds, and subsequently transferred the landmarks to both a 3D face mesh and a 3D virtual avatar with blendshape support.

As per the first research objective mentioned in section 1.3, our aim was to investigate and create a technique for transferring facial landmarks to a 3D face mesh. The research found that the developed plugin was able to successfully transfer facial expressions and movements from a person in a video input to a 3D face mesh. However, the facial landmarks predicted by MediaPipe were unstable and noisy, with jitter, resulting in inaccurate and blurry expression and movement transfer. Attempts to mitigate these issues using the 1€ filter were unsuccessful. Furthermore, the 3D face mesh height and width ratio did not match the input source face, and had to be manually tweaked for best results. This created unwanted 3D face mesh deformations at the cost of more human-like facial movements. To achieve more realistic facial movements while avoiding these issues, we propose capturing FACS Action Units, as depicted in fig. 2.5, instead of the entire range of facial movements. This approach needs to derive blendshape weights that matches the FACS Action Units, and apply them to an output 3D virtual avatar model with blendshape support. By doing so, we can avoid the issue of having deformed output 3D face mesh models that do not match the input face's geometrical width and height ratio.

In pursuit of research objective 2, as outlined in section 1.3, we implemented our proposed techniques (EAR/MAR) to determine the ratios of eye and mouth opening/closing. These ratios were then utilized to compute blendshape weights, which were subsequently transferred to a 3D virtual avatar with blendshape support. To visualize the blendshape weight calculations, we developed a tool that incorporated a webcam input feed and a trackbar for displaying the corresponding blendshape names. The findings revealed that we were able to successfully calculate the open/close blendshape weights for eye and mouth, which were applied to our 3D virtual avatar with success. However, our results only demonstrate this achievement for the most basic blendshapes, namely eyeBlinkRight, eyeBlinkLeft, and mouthOpen. To extend our research to more intricate blendshapes and animations, further investigation is required. To address this, we suggest generating a dataset consisting of facial landmark captures and blendshapes that can be used for machine learning purposes, with the ultimate aim of learning facial landmark features and predicting blendshape weights.

Chapter 4

Dataset creation for Neural Networks: The PointFaces Dataset

The current chapter will concentrate on objective 3 outlined in our problem statement as stated in section 1.3. As highlighted in the summary of 3.4, our aim is to identify an approach to generate multiple blendshape weights from facial landmark captures. In accordance with our objective, our goal is to produce a dataset that can be used in machine learning pipelines to predict blendshape weights from live video feeds, pre-recorded videos or images with human faces. The predicted blendshape weights obtained from this approach can be employed to animate a 3D virtual avatar. As far as we know, there was no comparable dataset available, which motivated us to produce our own dataset containing facial landmark data and 3D blendshape models.

The facial landmark data in our dataset was derived from Flickr-Faces-HQ (FFHQ), an image dataset of high-quality human faces originally designed as a benchmark for generative adversarial networks (GAN) [32], created by NVIDIA. We chose FFHQ as our dataset of facial images [39] due to its high resolution and large collection of human faces, which made it an ideal choice for our facial landmark capture experiments.

From the FFHQ image dataset, we selected a subset of 1000 facial images to create our own dataset with facial landmark capture data. We utilized the neutral face of the canonical model from MediaPipe shown in fig. 2.4b to produce our own 3D blendshape face models, which were deformations of the neutral face.

In the following sections, we provide details on the proposed pipeline that leverages the 3D blendshape face models and point cloud facial landmark capture data we produced, describing how we transform the captured facial landmarks using point cloud registration algorithms and how we create the 3D blendshape face models.

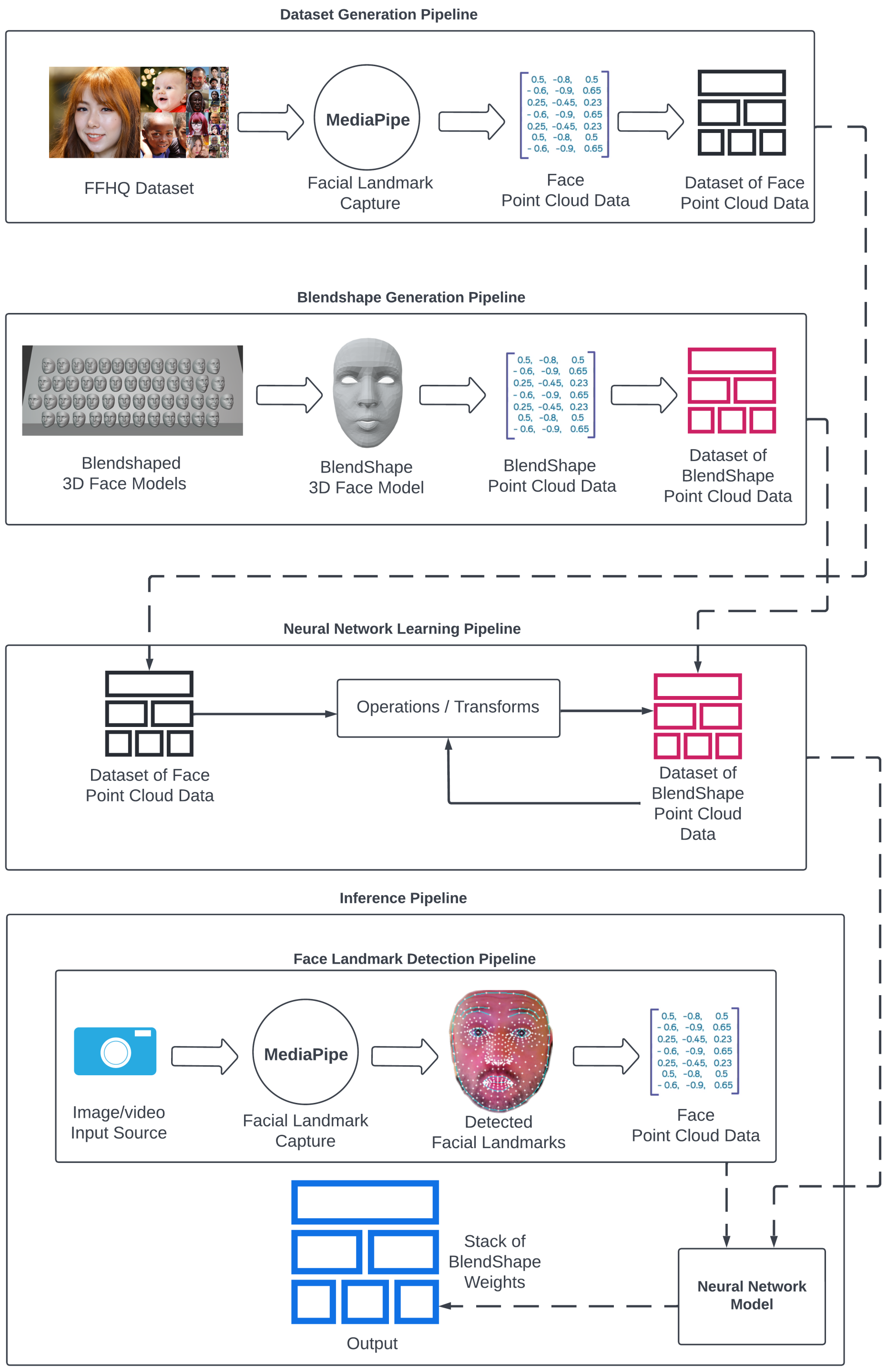
The dataset we have created consists of:

- Original FFHQ facial images (png).
- FFHQ facial images with facial landmark overlays (png).
- 3D point cloud data representing facial landmarks (ply/pcd/png).
- 3D point cloud data with transformation applied (ply/pcd/png).
- Sculpted 3D Blendshape face models based on the canonical neutral face model from MediaPipe (obj/ply/pcd/png).

Our dataset, which was named PointFaces, is published on GitHub, https://github.com/olealgoritme/pointfaces

4.1 Proposed Pipeline

Our proposed high-level pipeline involves utilizing point cloud data obtained from facial landmark captures on the FFHQ dataset and a set of 3D blendshape facial models, to train a model capable of learning blendshape weights that correspond to facial landmarks detected in input facial images, as part of machine learning pipelines. The pipeline we suggest, depicted in fig. 4.1, entails using MediaPipe to capture facial landmarks from human faces in videos, live feeds, or images. Subsequently, the obtained face point clouds are processed in a model trained on blendshapes and facial landmarks, to predict blendshape weights.



Dataset Generation Pipeline
FFHQ Dataset
MediaPipe
Facial Landmark Capture
Face Point Cloud Data
Dataset of Face Point Cloud Data
Blendshape Generation Pipeline
Blendshaped 3D Face Models
BlendShape 3D Face Model
BlendShape Point Cloud Data
Dataset of BlendShape Point Cloud Data
Neural Network Learning Pipeline
Dataset of Face Point Cloud Data
Operations / Transforms
Dataset of BlendShape Point Cloud Data
Inference Pipeline
Face Landmark Detection Pipeline
Image/video Input Source
MediaPipe
Facial Landmark Capture
Detected Facial Landmarks
Face Point Cloud Data
Stack of BlendShape Weights
Output
Neural Network Model

Figure 4.1: Proposed pipeline for blendshape weight prediction in neural networks.

4.2 Facial Images to Landmarks and Point Cloud Data

We extracted facial landmarks from input facial images in the FFHQ dataset, as shown in fig. 4.2, utilizing MediaPipe, OpenCV and Python3, as explained in sections 2.5, 2.9.1 and 2.9.5, respectively.

The facial landmarks we extract are saved in two distinct formats: as annotation overlays on top of the original images, as shown in fig. 4.3, and as geometry point clouds (x,y,z) in PCD/PLY file format[20, 49], as illustrated in fig. 4.4.

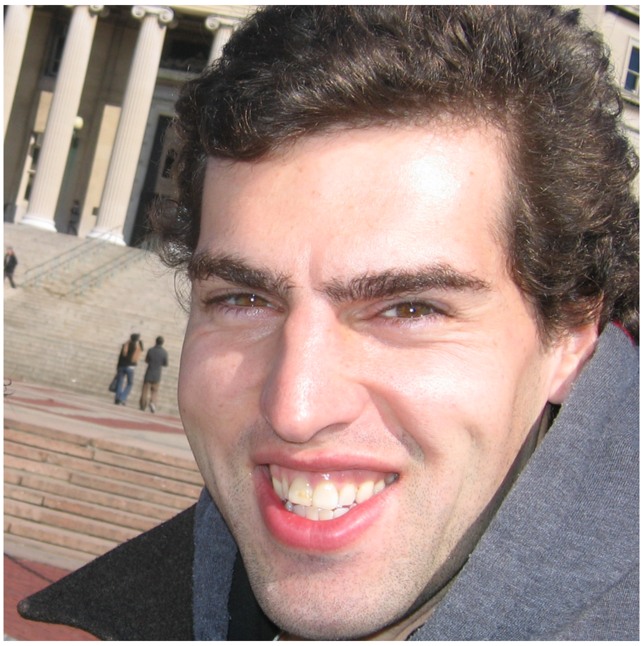


Figure 4.2: FFHQ Input image.

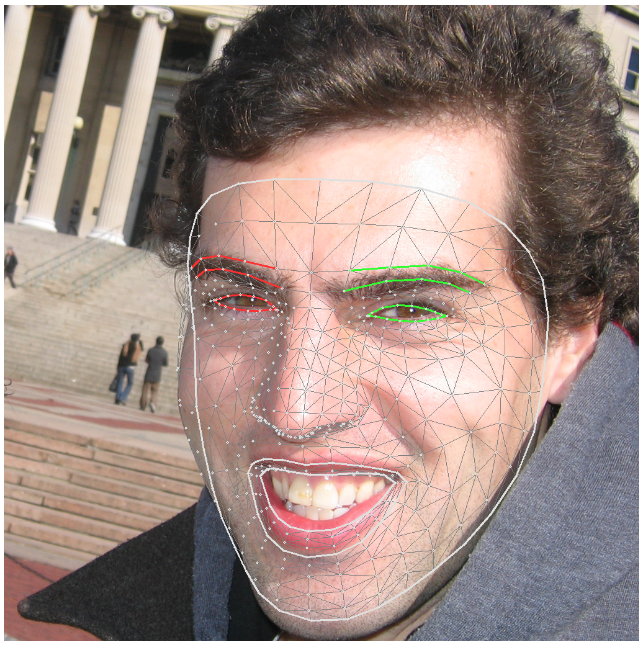


Figure 4.3: Facial landmark capture overlay.

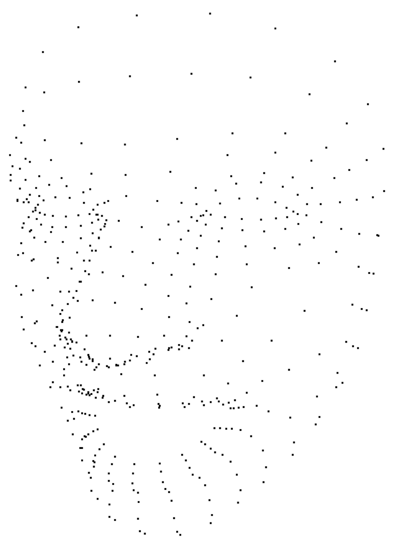


Figure 4.4: Point cloud of facial landmark capture without transformation applied.

4.3 Facial Transformation

Since the FFHQ human faces dataset includes images of faces with varying orientations, it was necessary to correct the orientation of the faces programmatically to ensure that all captured 3D face point clouds have their noses facing forward. The facial orientation can be observed in fig. 4.4, where the tip of the nose is pointing to the left instead of pointing forward. We correct this with head-pose-estimation - a non-trivial and unique challenge in computer science, which is normally used to minimize errors during machine learning training pipelines [51]. Finding the appropriate translation and transformation parameters to align the captured point cloud with the neutral face was a task that required a significant amount of time. As seen in fig. 4.5, the captured face is significantly smaller than the neutral face. Initially, we attempted to perform this alignment manually using Open3D, a library described in section 2.9.6, for scaling the captured point clouds to match the neutral face, as shown in fig. 4.6. We deem these steps a necessity to minimize errors when comparisons are performed between point clouds of blendshapes and point clouds of captured faces. However, manually rotating and scaling the point cloud is tedious and time consuming, so we we choose to employ point cloud registration algorithms instead.

We explore Arun's Method [6], which is described as "least-squares fitting of two 3D point sets" and is an alternative to ICP [18], for aligning our point cloud. Both ICP and Arun's method are techniques utilized for point cloud registration, which involves aligning one point cloud with another. ICP estimates closest points within N-dimensions between two point cloud sets, whereas, Arun's Method is a solution for finding the optimal rigid transformation between two point clouds. This process includes determining the singular value decomposition (SVD) of a matrix created by the covariance of two point clouds, and then using the resulting

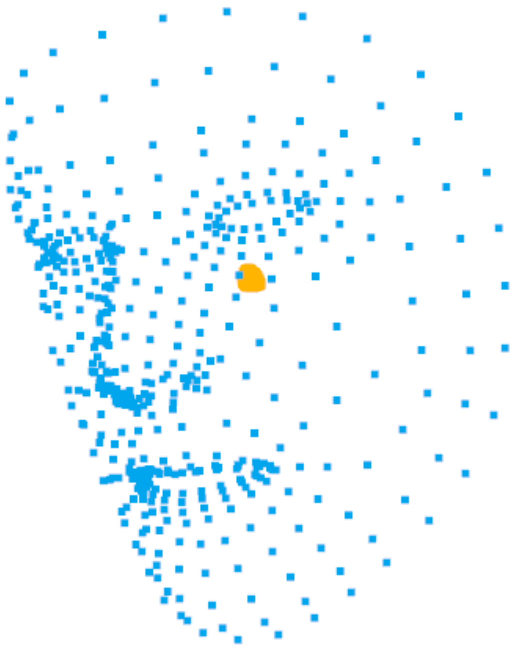


Figure 4.5: Neutral face (blue) and captured face (yellow) in their original form, without translation and transformation applied. The captured face is 25x smaller than the neutral face.

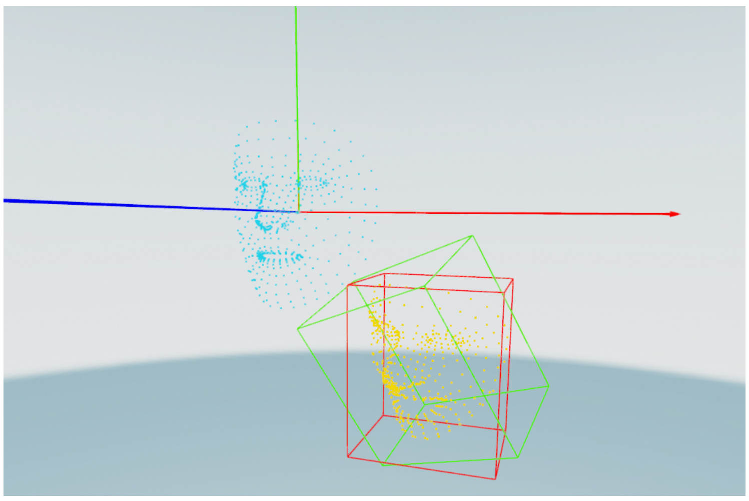


Figure 4.6: Neutral face (blue) and captured face (yellow) after manual rotation and scaling (25x) applied.

matrices to calculate the optimal rotation and translation parameters.

Using Numpy, described in section 2.9.4, with Python3 and Open3D, we utilized Arun's algorithm to process the captured point clouds. Arun's Method can also be implemented as a step in a neural network module to facilitate translation and rotation matching to a reference point cloud for other datasets. In our case, we apply this process directly to our dataset, rotating the point cloud of captured facial landmarks to align it with the rotation matrix, and translation of the point cloud representing the neutral blendshape (canonical face mesh), depicted in fig. 2.4b.

The results from performing rotation and translation on the captured face's point cloud to match the neutral face are presented in figs. 4.7 to 4.9. Our experimentation with Arun's Method for point cloud registration resulted in promising outcomes, which were satisfactory for our dataset.

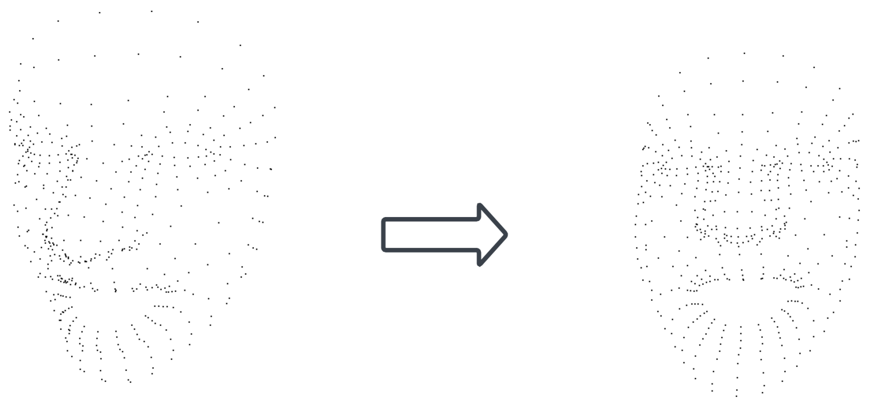


Figure 4.7: Left: Original Capture, Right: Transformed with Arun's Method to match the neutral face with nose pointing forward.

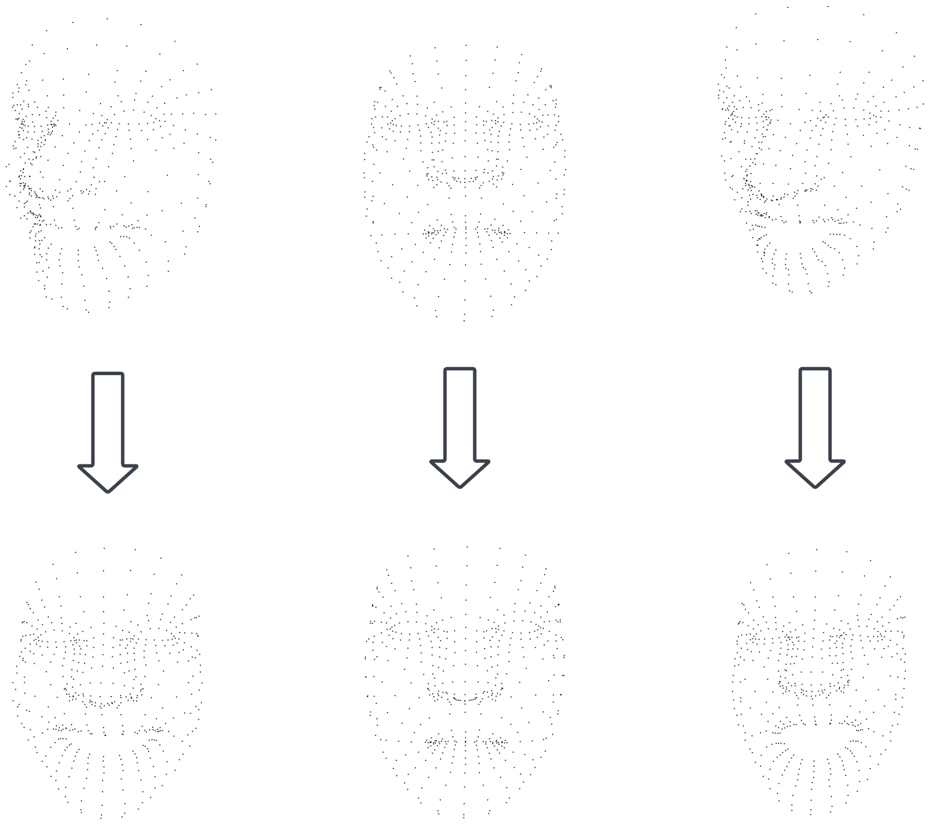


Figure 4.8: Top: Original Capture, Bottom: Arun's Method applied, final result.

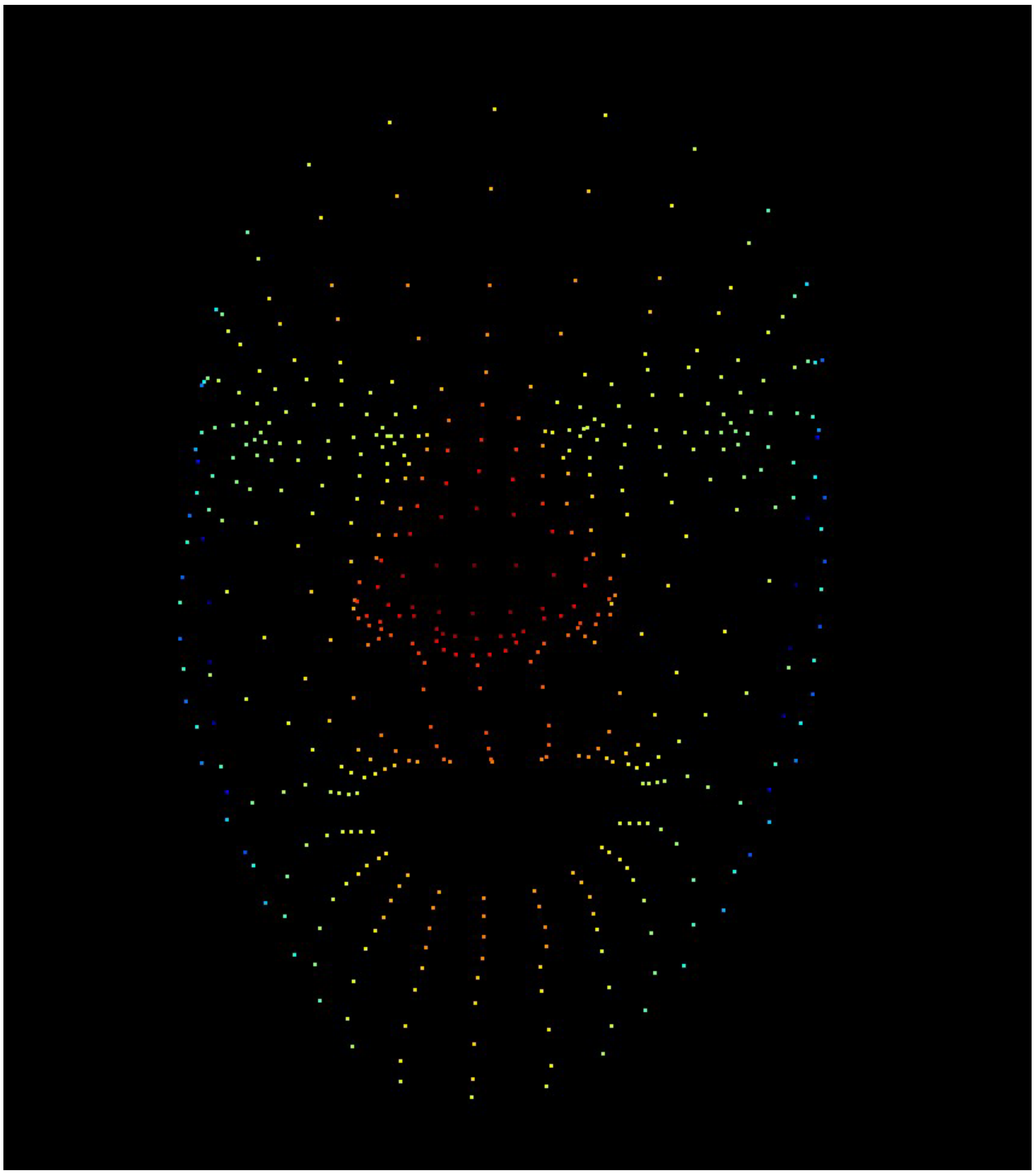


Figure 4.9: Facial landmark capture after Arun's Method is applied. Same landmark capture without the transformation is depicted in fig. 4.4.

4.4 3D Blendshape Face Models

To enable comparisons between the captured facial landmarks and their corresponding facial expressions, we sculpted a set of blendshapes. Blendshapes are a tool used in animation, described in section 2.6. The neutral face or neutral blendshape, illustrated as the canonical face model by MediaPipe in figs. 2.4b and 4.11, serves as our base model for creating all of the blendshapes.

The neutral face comprises of 468 vertices (points), and it is crucial that the 3D models of the blendshapes we create match the same number of vertices and vertex topology as the neutral face model. Non-matching vertex topologies between 3D blendshape models and the neutral face model makes it impossible for point-to-point comparisons/extractions. We highlight these issues in fig. 4.10.

To create each 3D blendshape face model, we use the models created by Apple as reference material, which are available at [5] as part of their ARKit. However, since the reference models have a vertex count of over 1200 and a different vertex topology from our neutral face model, we cannot use them directly. Instead, we manipulate the vertex positions of the neutral face model using Blender, a tool described in section 2.9.3, to create the corresponding blendshape models that match the ARKit models. We render the 3D models from ARKit in fig. 4.12 and create our blendshape models, seen in figs. 4.13 to 4.17.

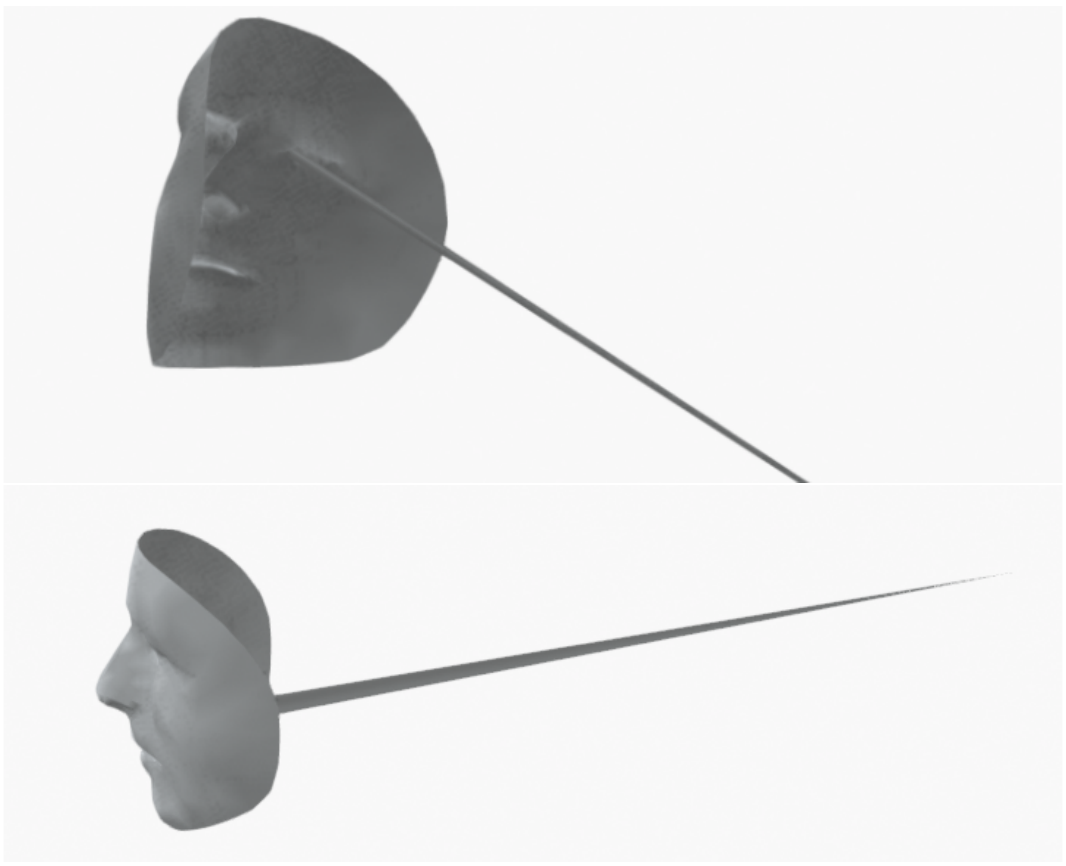


Figure 4.10: Illustrating how our sculpted 3D blendshape models (top) did not have the correct vertex topolgy when compared with the neutral face base model (bottom). The first indices of each face is extruded to highlight the first vertex.

.

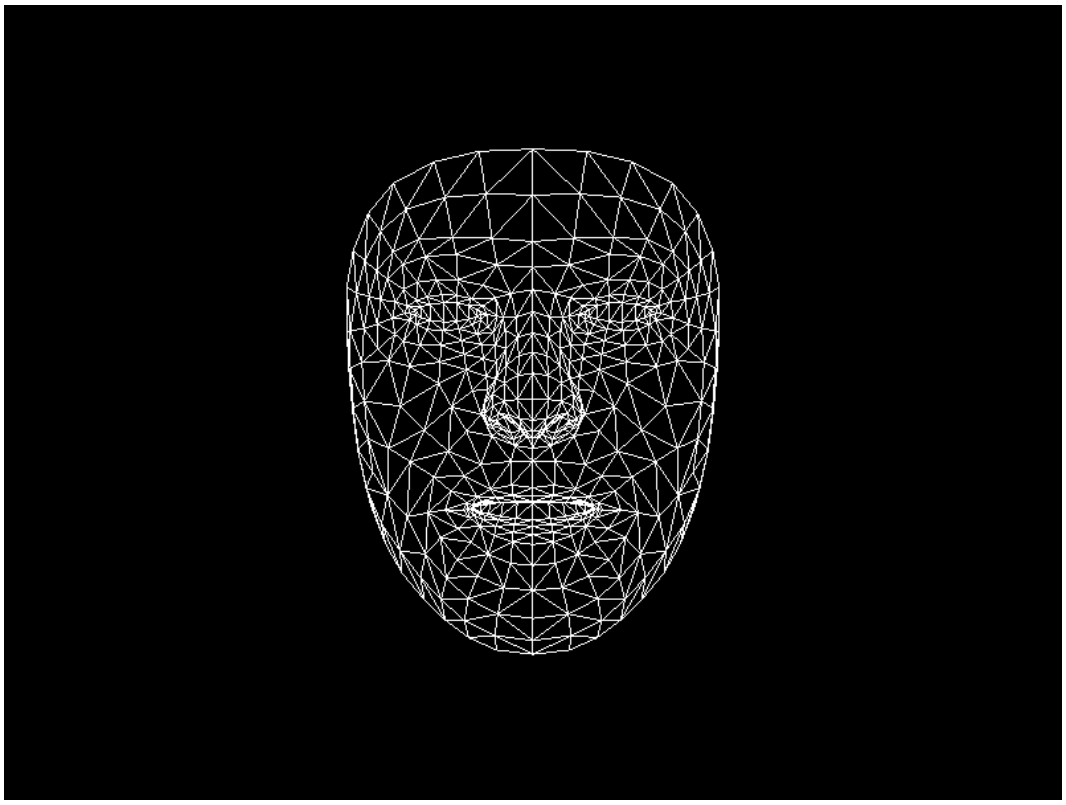


Figure 4.11: Our neutral face starting point. Rendered canonical face by MediaPipe[31] with 468 vertices. Lines are drawn between each vertex for a better understanding of how the model looks.

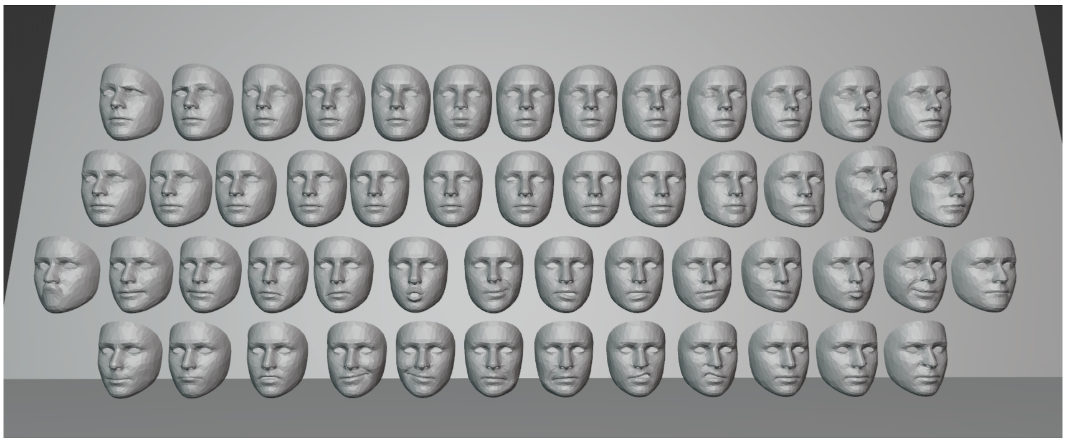


Figure 4.12: The ARKit, imported in Blender and lined up for presentation. Our blendshapes are based on these models. Each 3D model represents a Facial Action Unit [68].

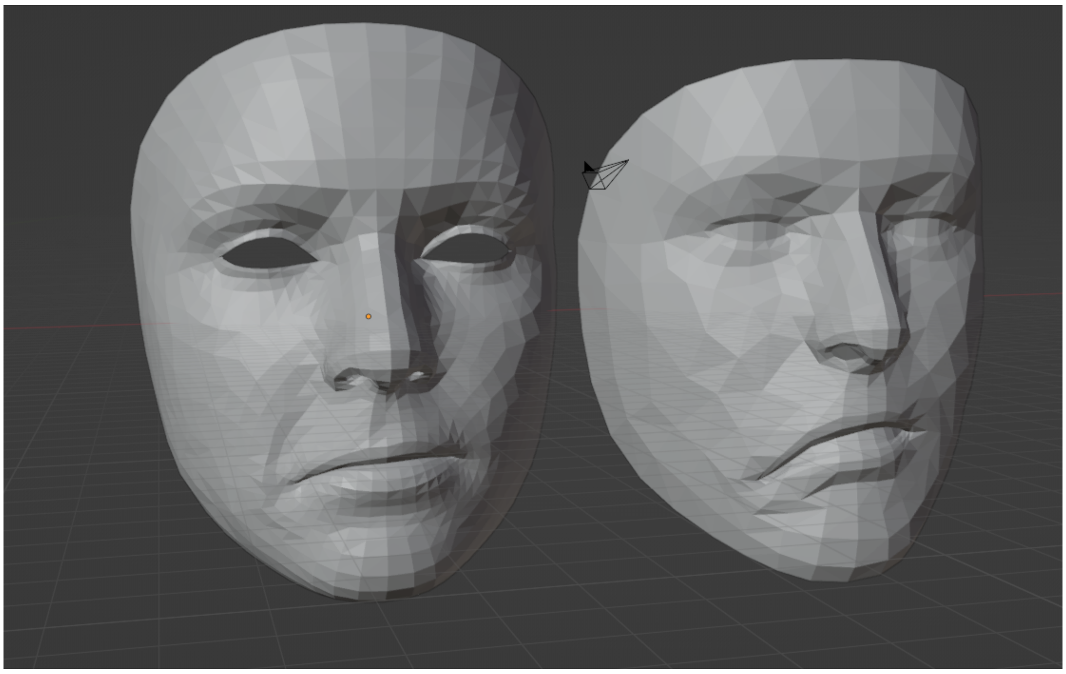


Figure 4.13: Left: ARKit Model, Right: Our model. Initial blendshape 3D Face creation process. This shows the inconsistencies and difficulties to produce similar facial expressions.

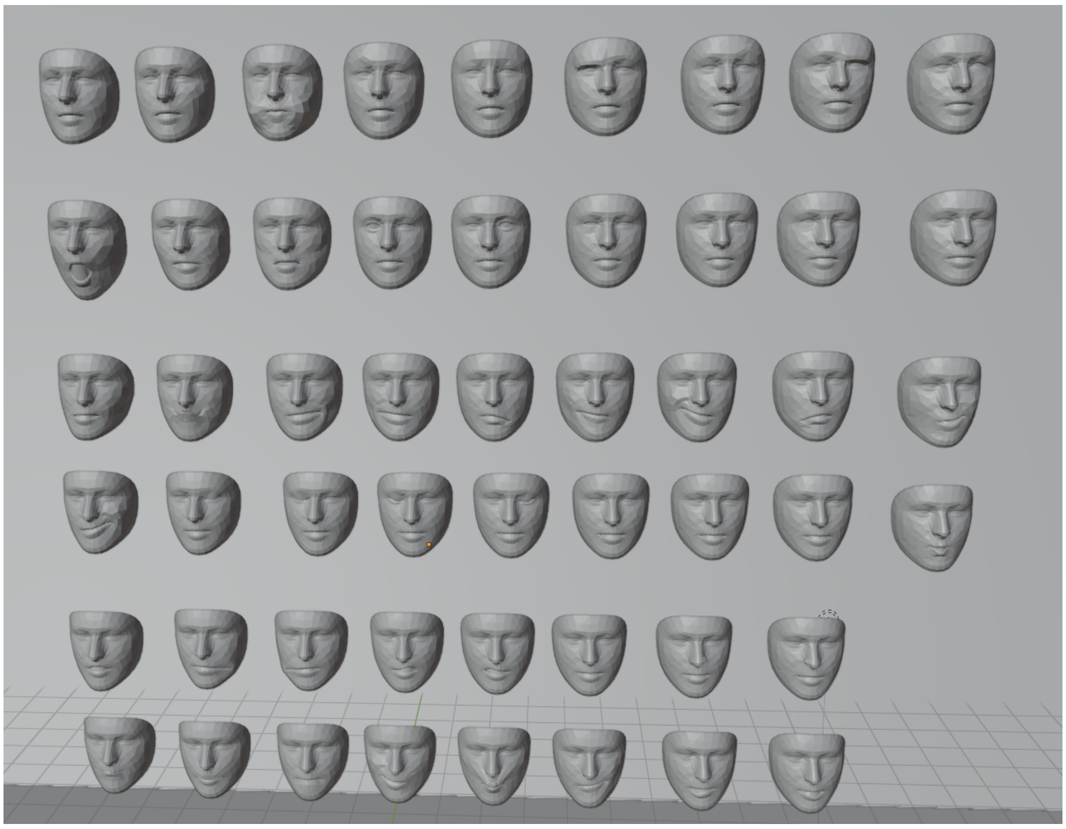


Figure 4.14: The finished blendshape lineup that was created, based on the Canonical Neutral Face. All models have the same vertex topology and vertex count matching the Canonical. Rendered in Blender.

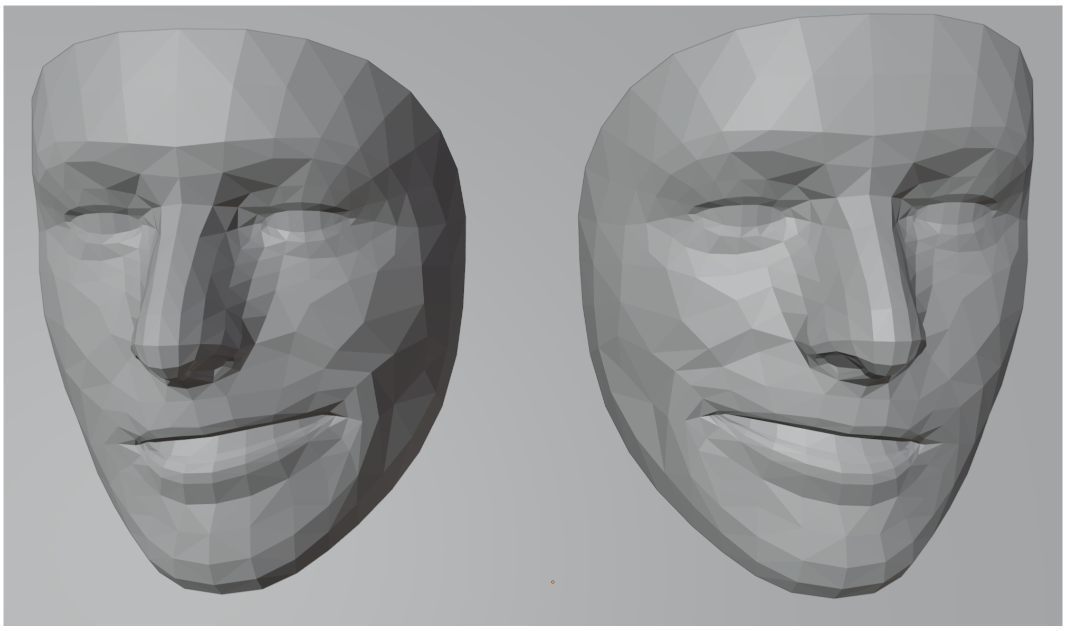


Figure 4.15: Left: mouthDimpleLeft, Right: mouthDimpleRight. Closeup look at the finished blendshapes. Rendered in Blender.

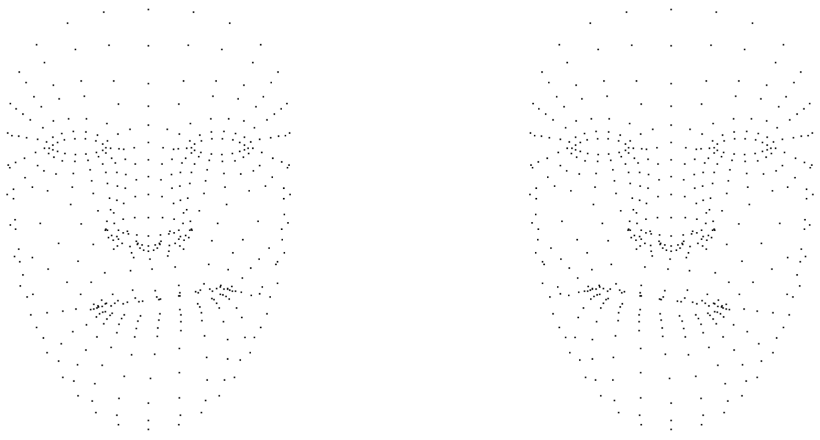


Figure 4.16: Left: mouthDimpleLeft, Right: mouthDimpleRight. The finished blendshapes shown as point clouds.

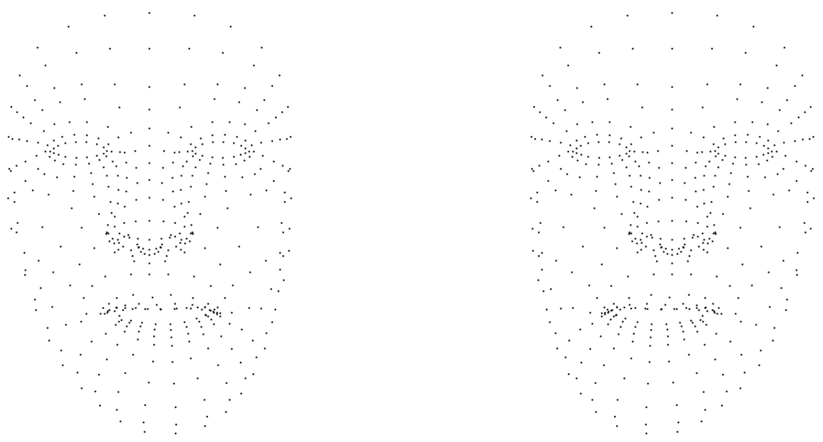


Figure 4.17: Left: jawLeft, Right: jawRight. The finished blendshapes shown as point clouds.

4.5 Summary

In this chapter we focused on producing a dataset that can be utilized in machine learning pipelines for blendshape weight prediction, in accordance with objective 3 in our problem statement stated in section 1.3. We created a dataset of facial landmarks in the form of image annotation overlays and face point cloud data, derived from the FFHQ dataset of facial images, accompanied with 3D blendshape face models.

To ensure minimal errors when using our dataset, the face point cloud data obtained from facial landmark captures had to be aligned with the translation, rotation, and scale of our base blendshape face model. Specifically, all of the face point cloud data had to have the nose pointing forward and be scaled to match the base blendshape face model. This was initially done manually but later automated with a point cloud registration algorithm known as Arun's Method, an ICP alternative.

While creating the 3D blendshape face models, we faced difficulties in automating the transfer of facial expressions from the ARKit source models to the models we were developing. This process caused a distortion in the vertex topology, moving the first vertex in our 3D model from the upper lip. As a result, we abandoned the automation process and instead opted for a manual approach, which produced the correct vertex topology and the results presented in this chapter.

Although creating our own dataset was a challenging and time consuming task, we met the requirements of our objective, which resulted in a collection of face point clouds obtained from facial landmark captures of the FFHQ facial images and sculpted 3D blendshape face models, each representing a unique blendshape. PointFaces, which we named our dataset, can be utilized for further research to explore the detection or prediction of various blendshapes from image or video inputs.

Chapter 5

Experiments with Machine Learning for Point Clouds

In this chapter, the emphasis is on the experiments conducted to implement machine learning architectures using the facial point cloud and blendshape dataset created in chapter 4, in accordance with objective 3 in our problem statement stated in section 1.3. The primary objective of this chapter is to assess the applicability of the dataset and its potential use in blendshape weight prediction. To accomplish this, we adopt a Siamese One Shot machine learning network influenced by [36, 58], a computer vision approach that distinguishes classification based on two separate images. Additionally, we explore the PointNet deep learning architecture for point cloud classification. The implementations detailed in this chapter are executed using Python and PyTorch, as explained in sections 2.9.1 and 2.9.2, combined with Jupyter Notebook for fast and interactive prototyping. We detail our findings and conclusions in the chapter summary.

5.1 Siamese One Shot Learning: Implementation Experiment

Siamese One Shot Learning is a computer vision method used to distinguish classification based on two separate images with feature mappings [42]. We selected this method as we aim to conduct straightforward comparisons between point clouds of our blendshapes and those of facial landmark captures.

As the name implies, a Siamese machine learning network consist of two identical networks, like depicted in fig. 5.1. Since the resulting machine learning weights are constrained to being identical for both of the siamese networks, we employ a single network and input two images in succession. Subsequently, we compute the loss value using a Contrastive Loss function and perform backpropagation.

5.1.1 Network Architecture

We base our implementation on the architecture in [36], a standard CNN. Our network is based on three consecutive CNN layers (96, 256, 384 kernels) using batch normalization for each convolutional layer (ReLU), followed by dropout. The network accepts 100x100 pixels and has three fully connected layers after the convolutional layers.

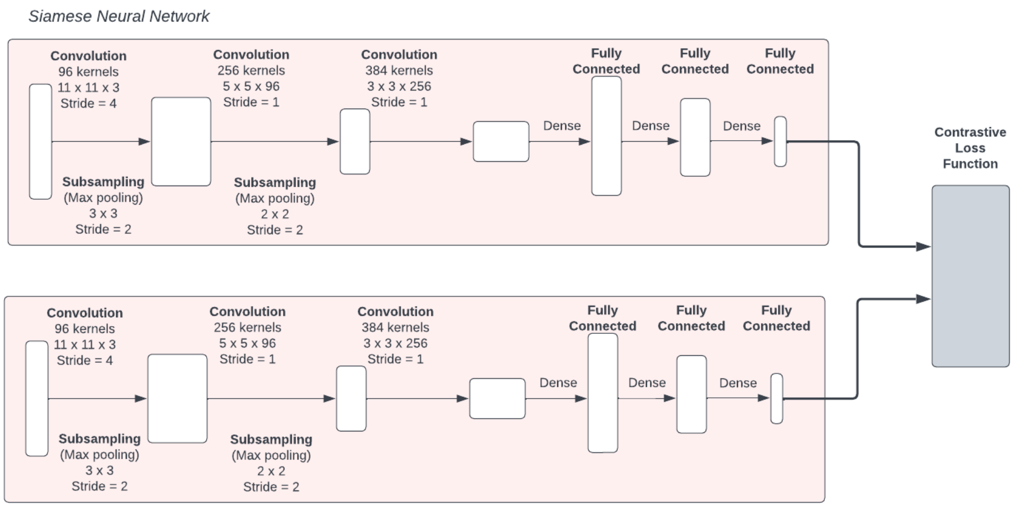


Figure 5.1: Simplied overview of a Siamese Neural Network Architecture.

5.1.2 Data Preparation

As our dataset consists of point cloud data from facial landmark captures and blendshape 3D models, which are stored as XYZ values in a text-based file, they cannot directly be processed for training by a Siamese One Shot Learning network, as it relies on RGB/monochrome data extracted from images. Thus necessitating the rendering and subsequent capture of the point clouds as images, as seen in fig. 5.5, a procedure conducted using Open3D, a tool previously detailed in section 2.9.6.

5.1.3 Training

The training of the Siamese One Shot machine learning network works by selecting an image pair as input, which in our case means an image of a blendshape and an image of a facial landmark capture. Each of these images are passed through the network, where the loss is calculated between the outputs of the two, using a loss function. The loss is backpropagated to calculate the gradient descents, and the weights are then updated using an optimizer. We utilize Contrastive Loss as our loss function, described in fig. 5.2, where *DW* is the euclidean distance and *GW* is the network output from a single image, depicted in fig. 5.3. The network is trained for 100 epochs, using Adam as our optimizer [15], and hyperparameters consisting of learning rate 0.0005. The results are shown in fig. 5.6.

$$(1-Y)\frac{1}{2}\left(D_W\right)^2+(Y)\frac{1}{2}\left\{\max\left(0,m-D_W\right)\right\}^2 \tag{5.1}$$

Figure 5.2: Contrastive Loss Formula

$$\sqrt{\left\{G_W\left(X_1\right)-G_W\left(X_2\right)\right\}^2} \tag{5.2}$$

Figure 5.3: Euclidean Distance Formula

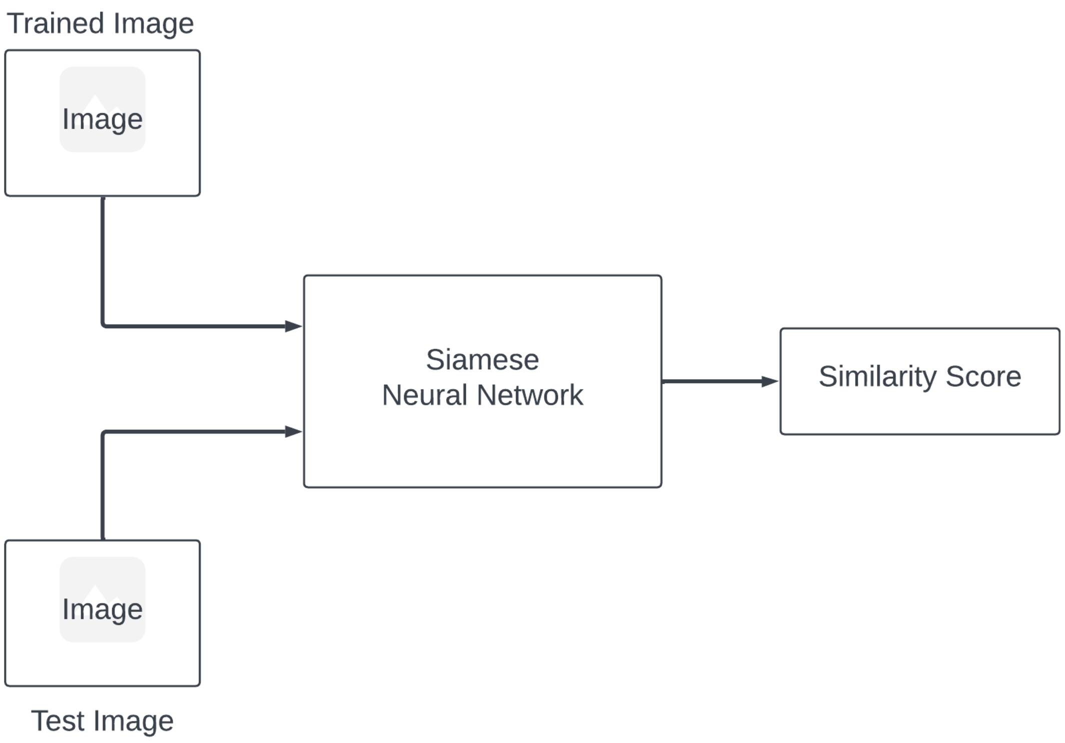


Figure 5.4: Simplified overview of how a Siamese One Shot Learning Neural Network works.

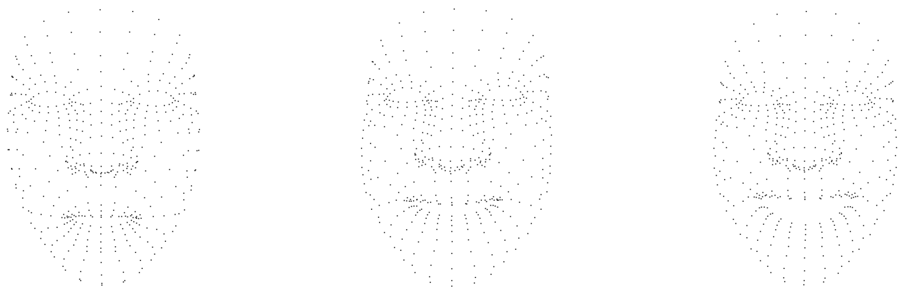


Figure 5.5: Captured facial landmark point clouds in the form of images.

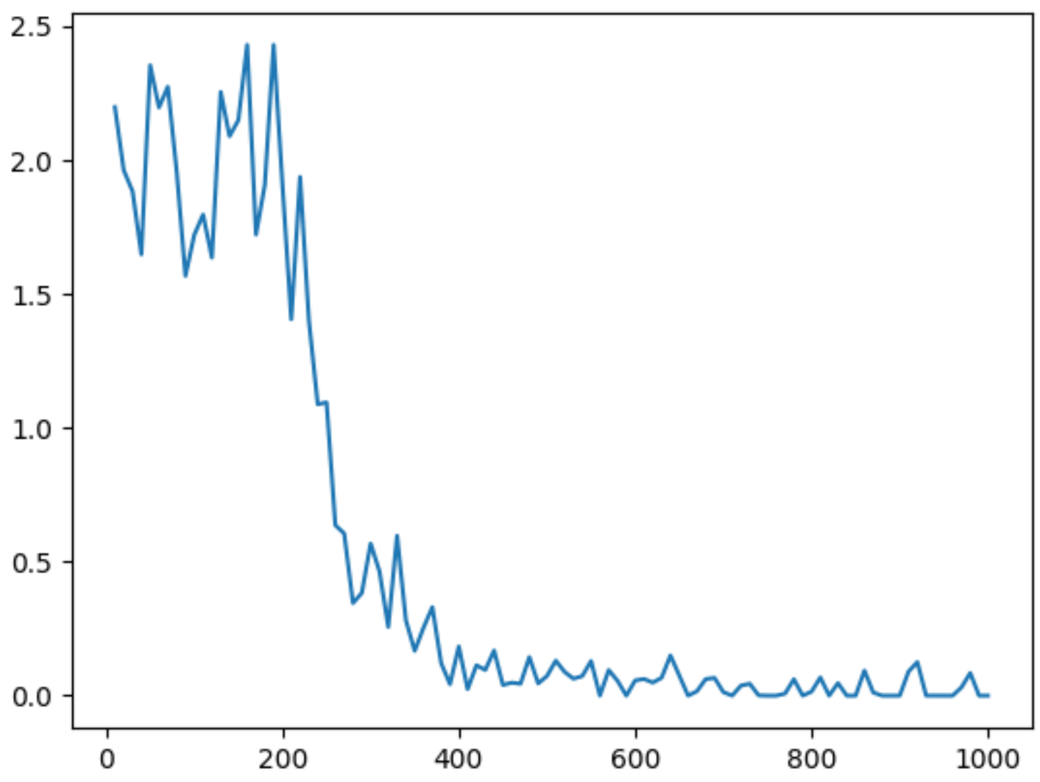
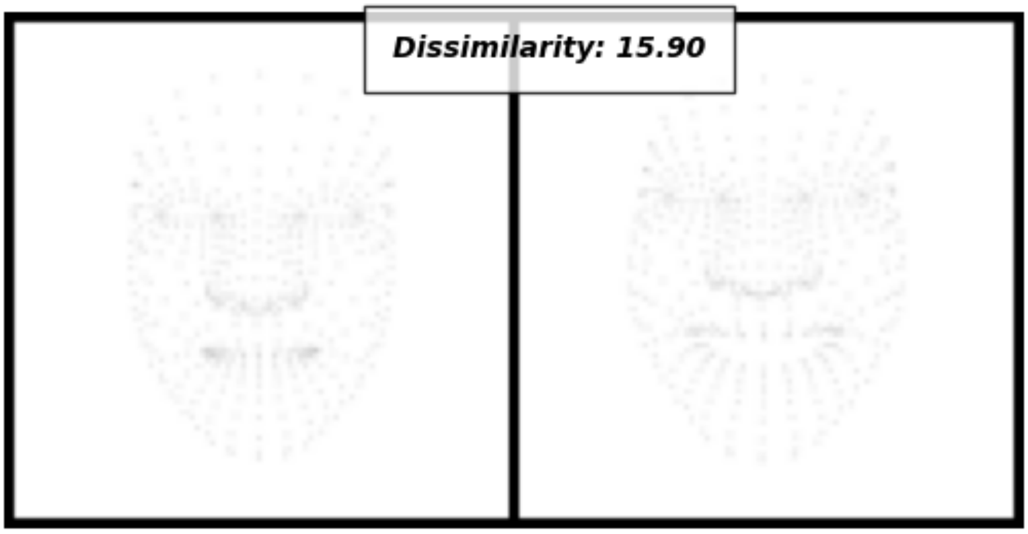
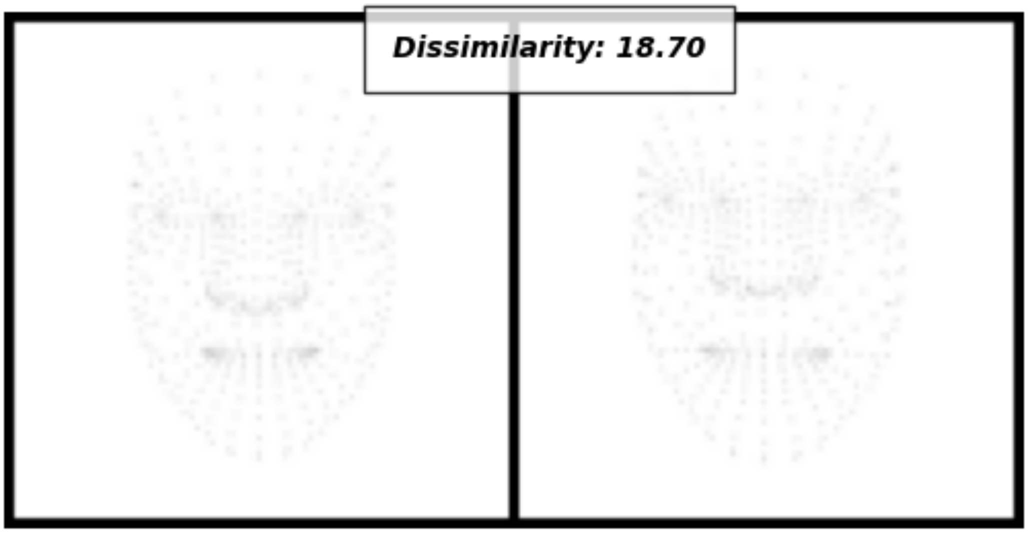
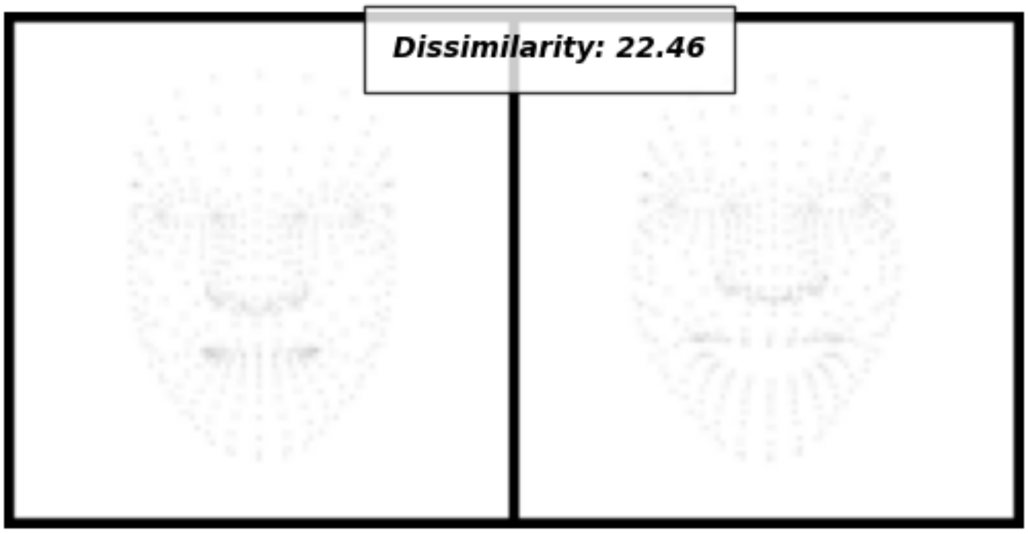
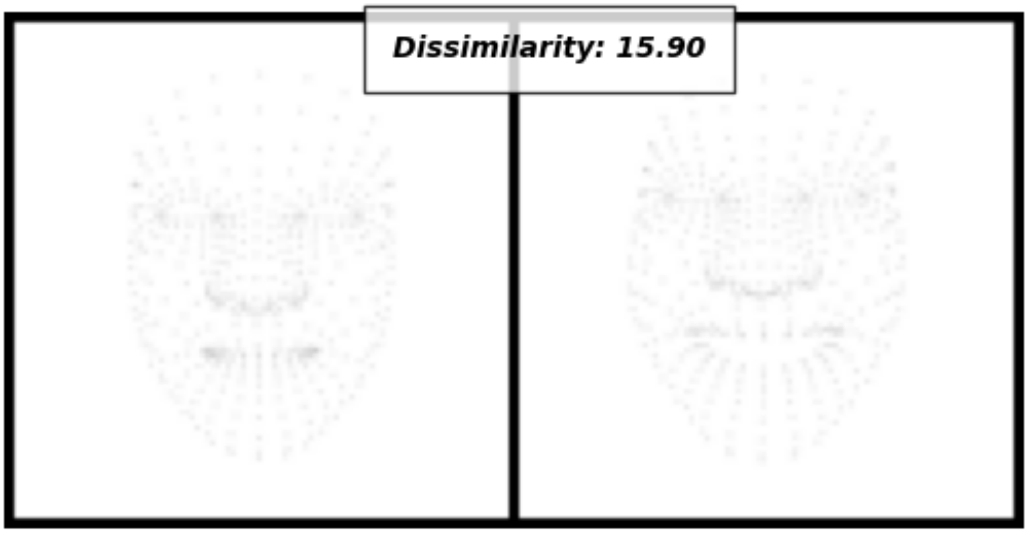
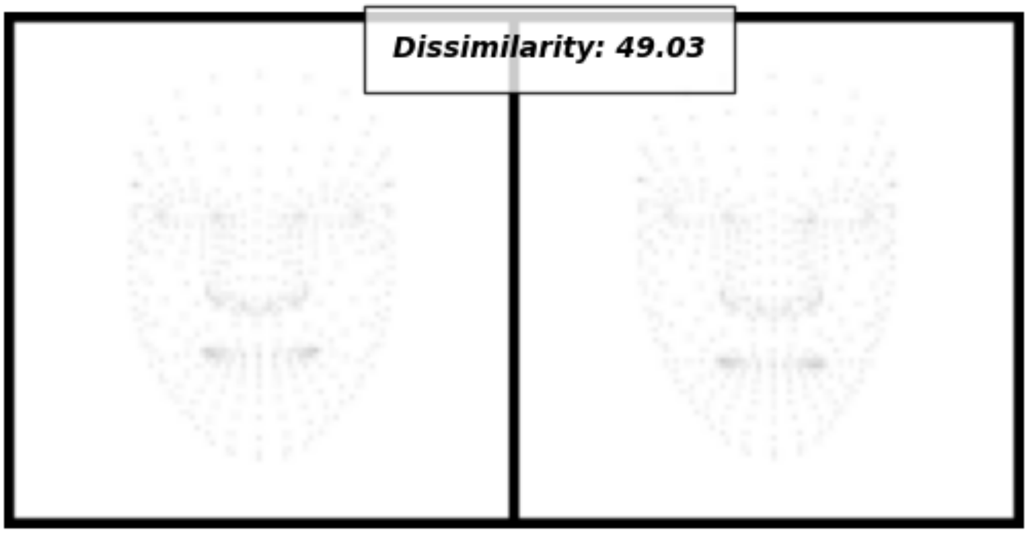
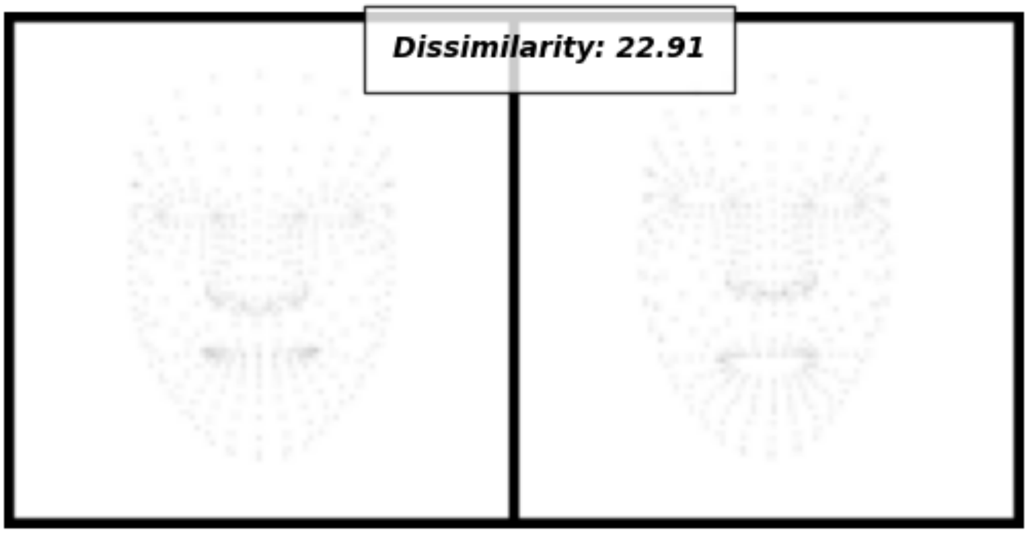


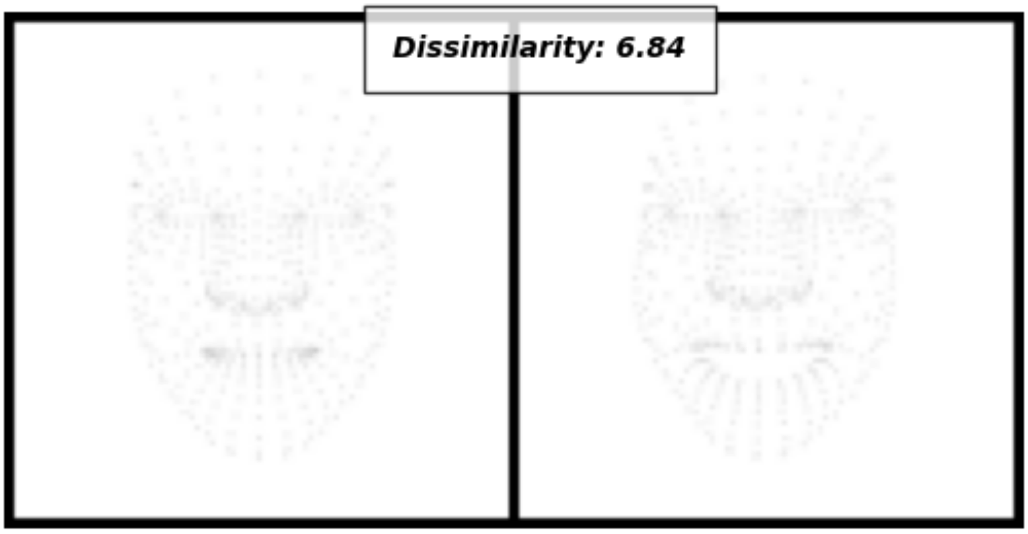
Figure 5.6: Contrastive Loss function results after 100 epochs. Graph depicts loss over time.

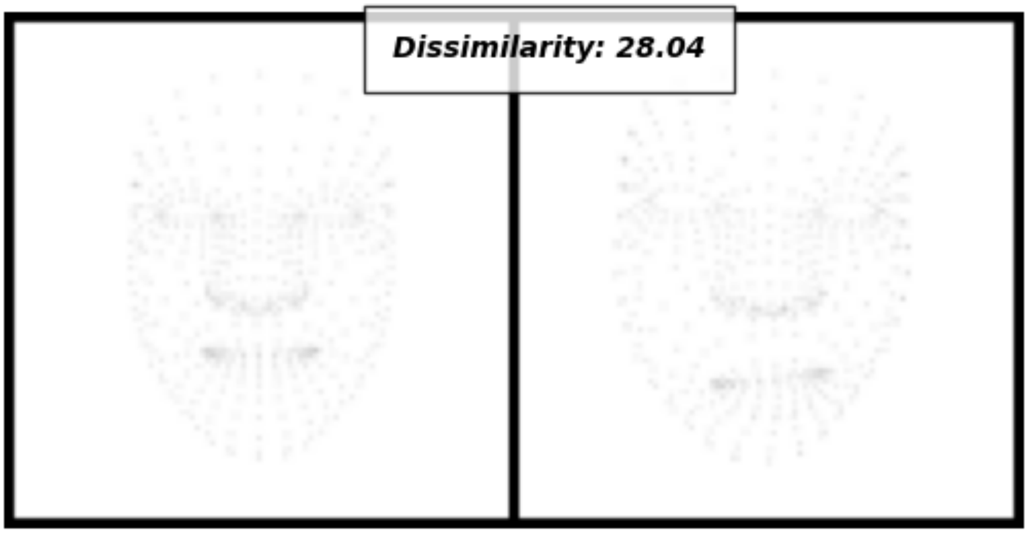

Dissimilarity: 15.90

Dissimilarity: 18.70

Dissimilarity: 22.46

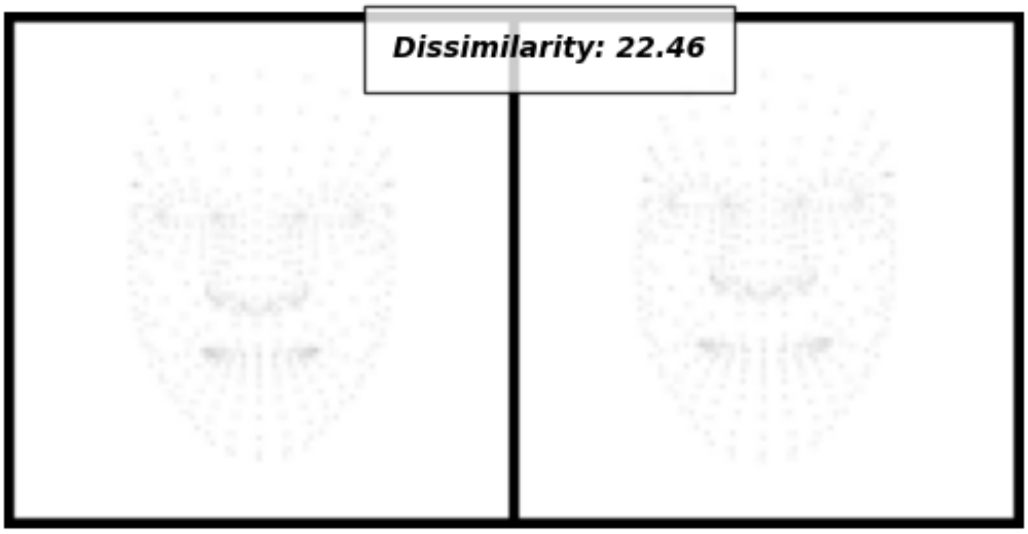

Dissimilarity: 15.90


Dissimilarity: 49.03


Dissimilarity: 22.91


Dissimilarity: 6.84


Dissimilarity: 28.04


Dissimilarity: 22.46

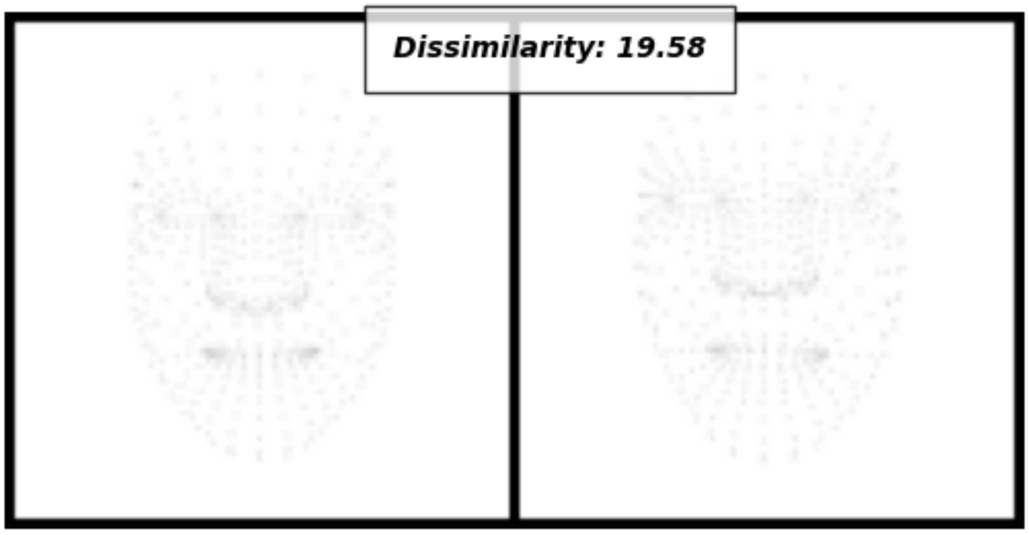


Figure 5.7: Dissimilarity results after running 100 epochs with Siamese One Shot Learning. This test is performed using the captured facial images as training images, and the blendshape images as the testing images. 10 images were extracted from the training images and compared to a random blendshape image.

5.2 PointNet: Implementation Experiment

PointNet: a state-of-the-art point cloud deep learning network, developed by Stanford University in 2017 [13]; used for point cloud data in 3D classification, segmentation and semantic scene parsing [57].

5.2.1 Network Architecture

The machine learning model for classification takes in n points as its input, applies input and feature transformations, and subsequently aggregates point features by utilizing max pooling. The output generated is the classification score for m classes. On the other hand, the segmentation network is an expansion of the classification network, which concatenates global and local features and outputs scores for each point. In the multi-layer perceptron (MLP), which is represented by the numbers in the bracket, batch normalization is used for all layers with Rectified Linear Unit (ReLU), similar to the normalization steps done using the Siamese network, as seen in section 5.1.1. Furthermore, dropout layers are integrated into the final MLP of the classification network.

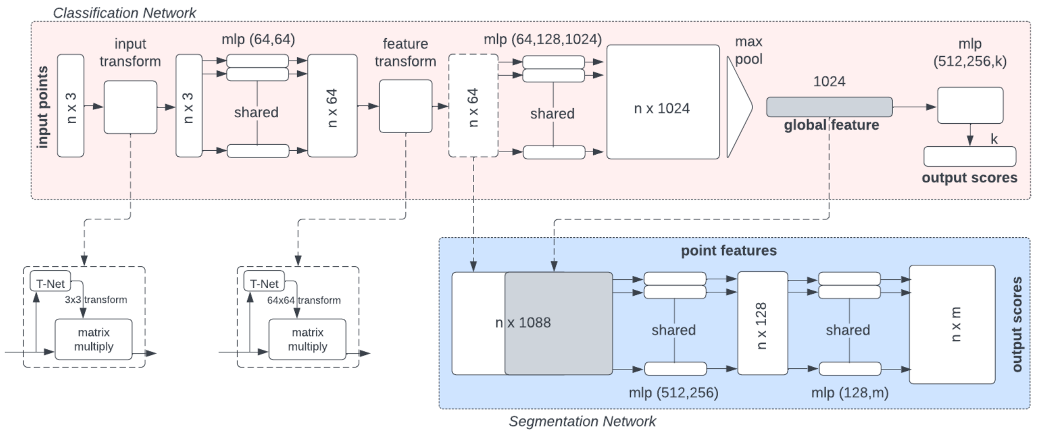


Figure 5.8: The PointNet Architecture.

5.3 Data Preparation

During the initial testing phase, it became evident that our point cloud data needed to have both vertex data and normals [19] and faces [17]. Unfortunately, our facial landmark capture data only consisted of vertex points and was missing both normals and faces.

To prepare the data for PointNet, we pre-processed our point cloud data, attempting to create triangle meshes that were needed for PointNet. We first estimated normals for our vertex points utilizing the method shown in fig. 5.10 and then experimented with surface reconstruction algorithms using the resulting points and normals.

We experimented with several surface reconstruction algorithms, such as Alpha Shapes [27], Poisson [40], and Ball Pivoting [10], in an attempt to reconstruct a triangle mesh representing the subject's face. Of these algorithms, Ball Pivoting produced the best results, yet still had missing triangles in the resulting face mesh, as illustrated in fig. 5.11. These results were not satisfactory enough for further processing.

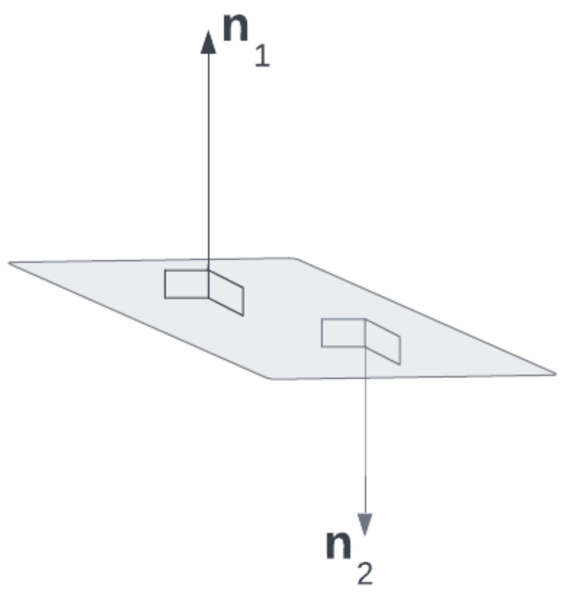


Figure 5.9: Simplistic view of what 3D normals looks like. n_1 and n_2 are 3D normals, pointing upwards and pointing downwards, respectively.

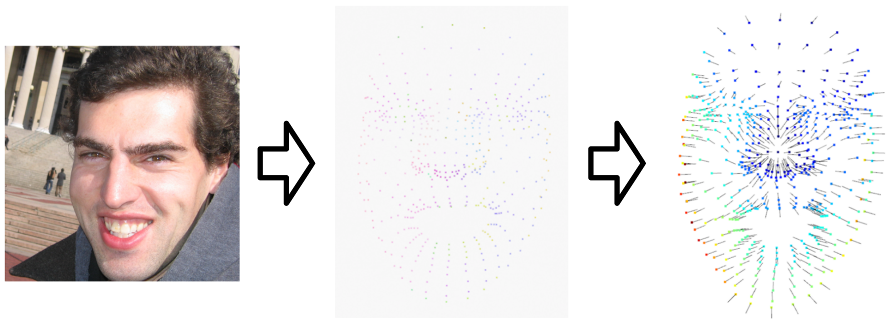


Figure 5.10: The 3D normals computation flow. From input image, to point cloud with vertex data, to 3D normals.

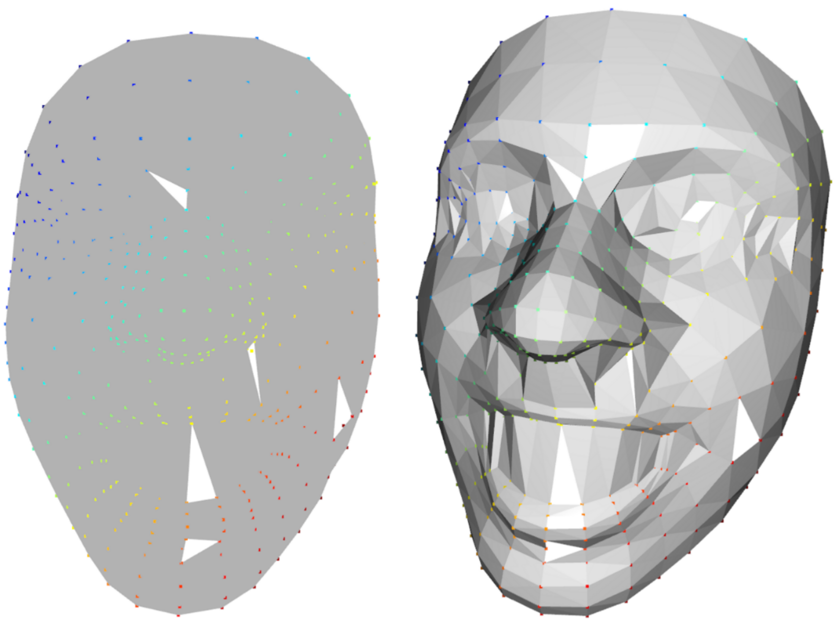


Figure 5.11: Face reconstruction results using Ball Pivoting. The parts in white represent the missing triangles after reconstruction.

5.4 Summary

This chapter focused on implementing machine learning architectures using PointFaces: the dataset we created in chapter 4, in accordance with objective 3 in our problem statement stated in section 1.3. We experimented with data pre-processing for a Siamese One Shot Learning network, and PointNet, a state-of-the-art deep learning architecture. In this summary, we conclude some observed results for both Siamese and PointNet.

Siamese One Shot Learning

- When comparing blendshapes with captured faces, we got results that could identify if two faces were exact matches giving us a Dissimilarity score of 0, or a Dissimilarity score that was difficult to interpret.
- This specific type of training is primarily used with images, so not an ideal method for getting accurate results based on point clouds. For us to experiment with Siamese One Shot learning we had to render point clouds and convert the render into an image that could be used for Neural Network training and testing.
- Captured faces needed facial transformations using the ICP algorithm to transform the faces to nose-forward, matching the blendshapes.

PointNet

- We focused on the experimentation of utilizing our dataset with the PointNet deep learning architecture. We discovered that our dataset needs pre-processing, and employ different techniques to reconstruct our data into triangle meshes.
- The results showed that the normals computation were partly accurate, but had missing normals. The resulting point cloud containing the computed normals were passed on to the Ball Pivoting 3D face re-

construction algorithm, which resulted in missing triangles in the triangle mesh, as indicated by the facial reconstruction images depicted in fig. 5.11.

- We can clearly see the resulting gaps indicated by the missing triangles. Our conjecture was that we could restore the missing normals and faces of the 3D point cloud to be compatible with the PointNet framework, which is considered a state-of-the-art framework for 3D point cloud data.

We chose to not further pursue the PointNet implementation, as the data pre-processing would be both time consuming and with uncertain results. We conclude that we need a more robust approach to get the results we are looking for, as the Siamese One Shot Learning network is aimed towards learning where training data is limited [42]. Siamese One Shot Learning results show us if two faces are dissimilar, with hard to interpret results, where as we need to determine how much a captured face resembles two other faces (neutral and blendshape faces) on a scale between 0 to 1. We suggest taking a closer look at specific operations and algorithms for point cloud comparisons, and do experiments with point cloud specific machine learning architectures.

Chapter 6

Implementations and Experiments with Point Cloud Distance Algorithms for Blendshape Weight Estimation

This chapter is devoted to exploring the methodology for quantifying the similarity between the facial expressions found in facial landmark captures and a neutral and fully expressed blendshape, using distance measuring algorithms on point cloud data. The measure of facial expression similarity is obtained between three different faces, namely, a neutral face, a captured face, and fully expressed face, and is expressed as a blendshape weight.

We propose that to leverage datasets of facial landmark captures and blendshapes, like we created in chapter 4, we must annotate each individual face in the dataset with labels that correspond to their facial expressions as blendshape weights, for the purpose of blendshape weight prediction in supervised machine learning pipelines.

In this chapter, we report the effectiveness of utilizing distance measuring algorithms on point clouds for such automatic annotation of facial images with computed blendshape weights that match their respective facial expressions. This process holds the potential to automate the time consuming process of data annotation of facial expressions for large scale facial image datasets, which is described as objective 4 in our problem statement stated in section 1.3. Ultimately, having large datasets with pre-annotated blendshape weights could lead us to our overarching goal, namely, the development of a machine learning model trained to compute blendshape weights from facial images captured in videos or live camera feeds, which can be applied to any blendshape supported 3D avatar for realtime avatar animation.

We propose a method where we can measure the distance between the points in facial point clouds to determine how much of a similarity there is between point clouds of a captured face, a neutral blendshape and a target blendshape, indicating the blendshape weight matching the facial expressions of a captured face.

This proposed method is based on our findings when applying the K-D Tree Nearest Neighbor algorithm, as described in section 6.2.1, for distance measuring, showing the co-relation of point distances between the neutral blendshape seen in fig. 6.2a and a target blendshape, as seen in fig. 6.2b. The distances between individual points were measured between the neutral blendshape and blendshapes mouthSmileLeft and mouthSmileRight, seen in figs. 6.2b and 6.2d. The highlighted points between 40-80 in fig. 6.2c show that the distances was found to be greater in fig. 6.2b, indicating a left smile in the blendshape. The distance between points 0-40 shown in fig. 6.2e was found to be greater in fig. 6.2d, indicating a right smile in the blendshape.

Based on our visual similarity findings, we explore our methodology for blendshape weight calculations. We selected a subset of facial captures from the dataset we created in chapter 4, for analysis using distance measuring algorithms on facial points clouds. Specifically, our focus shifted towards the mouth area and its corresponding blendshapes, not including the entire face, due to the complex nature of facial expressions. In the following sections, we present the methods and results obtained by applying these algorithms to a single blendshape, mouthSmileLeft, depicted in fig. 6.2b, and then extend our analysis to applying computations of blendshape weights using 24 different mouth-related blendshapes seen in fig. 6.21. We discuss our findings and evaluate the effectiveness of the algorithms employed in the chapter summary.

For all subsequent blendshape weight computations within this chapter, we apply the following equation:

$$Weight = D(N, B) - (D(C, N) + D(C, B)/2) \tag{6.1}$$

where D is the point cloud distance algorithm applied, N is the neutral blendshape, B is the target blendshape and C is the captured face. A simplified visualization of how the blendshape weight is obtained is depicted in fig. 6.1.

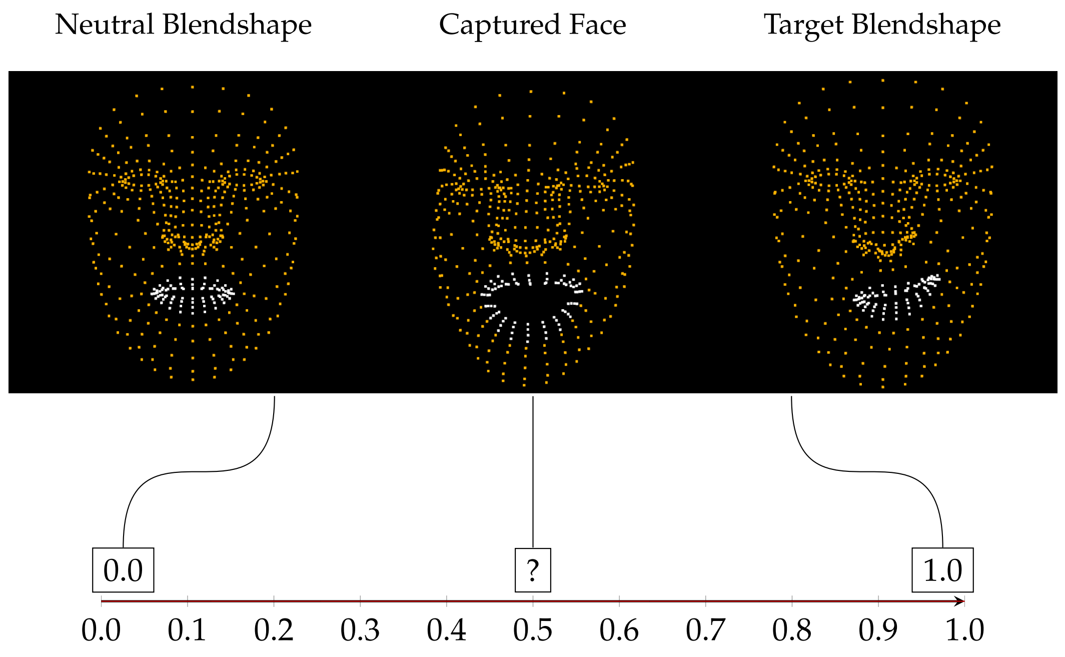
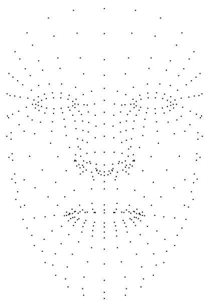
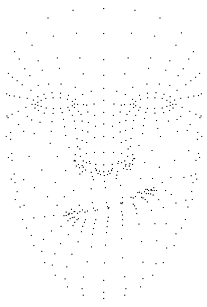


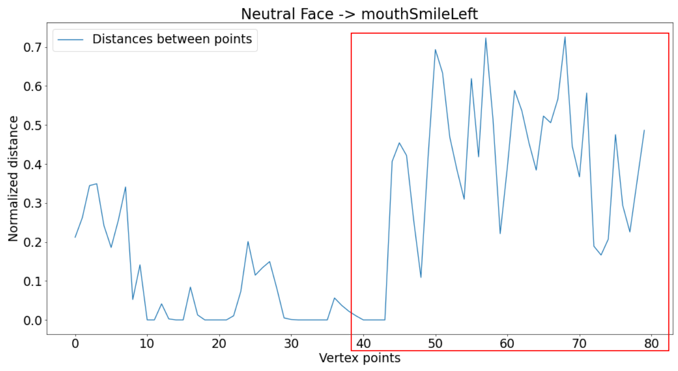
Figure 6.1: A visualization of how blendshape weight calculation is done for a captured face. Specifically, the computed blendshape weight value for a corresponding facial expression will range between 0 and 1. In this example, the target blendshape is mouthSmileLeft, as depicted in fig. 6.2b.



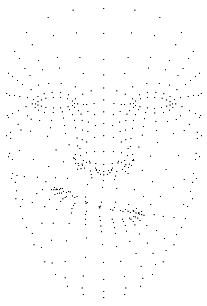
(a) Neutral blendshape



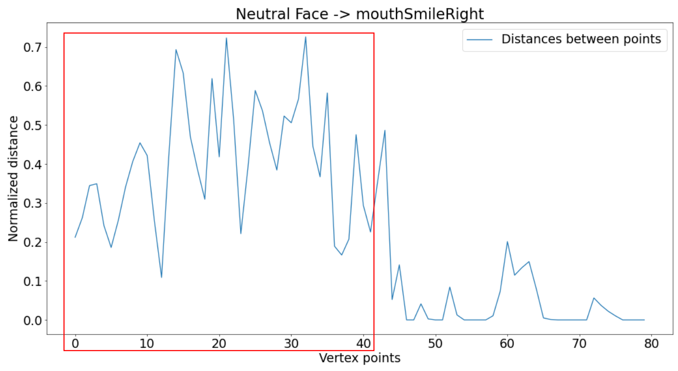
(b) mouthSmileLeft blendshape



(c) Individual point distances between neutral blendshape seen in fig. 6.2a and blendshape mouthSmileLeft seen in fig. 6.2b.



(d) mouthSmileRight blendshape

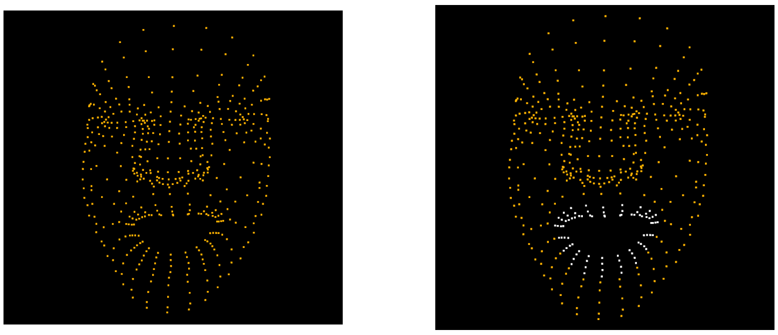


(e) Individual point distances between neutral blendshape seen in fig. 6.2a and blendshape mouthSmileRight seen in fig. 6.2d.

Figure 6.2: Visualization of the co-relation between individual points of neutral blendshape seen in fig. 6.2a and blendshapes mouthSmileLeft and mouthSmileRight seen in figs. 6.2b and 6.2d respectively.

6.1 Extracting the mouth points

The first step in our distance measurement application involve the extraction of the mouth region, which is characterized by a specific set of geometry points. This step is taken to enhance the precision of the measurements, as our focus is primarily on the mouth area, and other facial features are irrelevant to our analysis. From the MediaPipe facial landmark capture, our resulting facial point cloud comprises a total of 468 points, with 80 points specifically related to the mouth area, as depicted in fig. 6.3. The mouth region of 80 points is extracted from the facial point clouds of a captured face, neutral and target blendshape. Subsequently, the distances between the three extracted mouth regions of captured face, neutral and target blendshape are computed to obtain a blendshape weight.

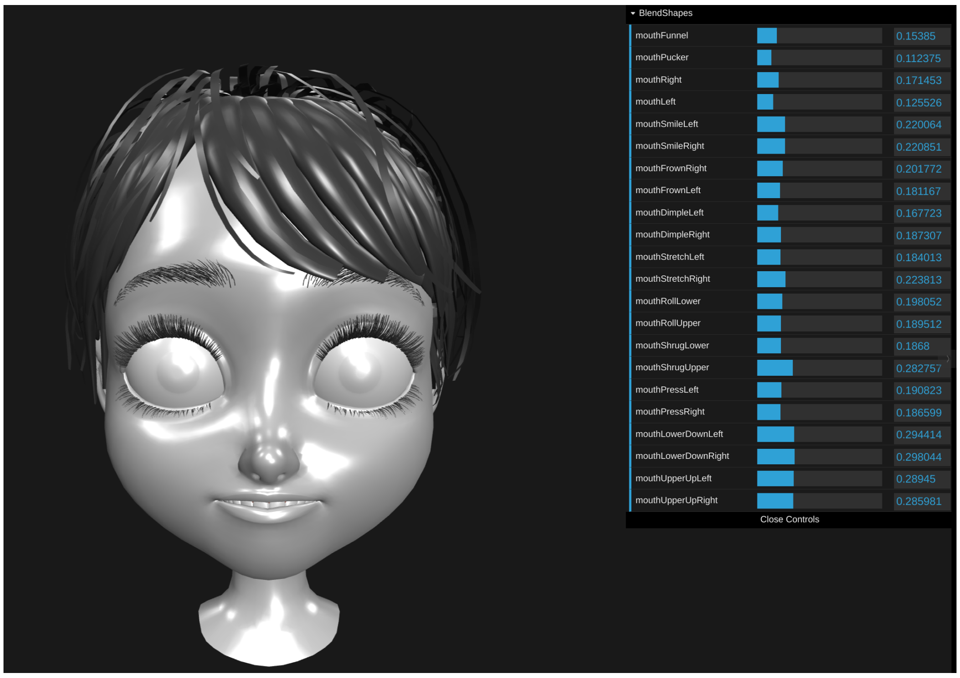

(a) Point cloud of a facial landmark capture.

(b) Highlight of the 80 point mouth area.

Figure 6.3: Mouth points to be extracted for distance measuring between a neutral and target blendshape.

6.2 The Model Viewer: Web Based Blendshape Animator

As we saw the need to visualize and evaluate the results of the blendshape weights on a 3D avatar, we create our own web based model viewer, a fork of the glTF-viewer [50] from threejs, a 3D JavaScript library [16]. The model viewer can view any type of glTF 3D model, and we use our previously used polywink face model with blendshapes, after a conversion from the original FBX model to a compatible glTF model. The viewer is hosted locally and accepts HTTP requests along with URL parameters specifying vectors of blendshape names and blendshape weights to be applied to the avatar.



BlendShapes

mouthFunnel	0.15385
mouthPucker	0.112375
mouthRight	0.171453
mouthLeft	0.125526
mouthSmileLeft	0.220064
mouthSmileRight	0.220851
mouthFrownRight	0.201772
mouthFrownLeft	0.181167
mouthDimpleLeft	0.167723
mouthDimpleRight	0.187307
mouthStretchLeft	0.184013
mouthStretchRight	0.223813
mouthRollLower	0.198052
mouthRollUpper	0.189512
mouthShrugLower	0.1868
mouthShrugUpper	0.282757
mouthPressLeft	0.190823
mouthPressRight	0.186599
mouthLowerDownLeft	0.294414
mouthLowerDownRight	0.298044
mouthUpperUpLeft	0.28945
mouthUpperUpRight	0.285981

Figure 6.4: The Model Viewer showing the polywink 3D face model with multiple blendshape weights applied to the avatar.

6.2.1 K-D Tree Nearest Neighbor Distance Algorithm

K-Dimensional trees or K-D trees, are binary search trees represented as a set of N points in K-dimensional space, in our case 3D space. Many kinds of searches can be performed on a K-D tree, including exact-match, partial-match, and range queries, which are often used in database applications due to it's often tree-like, complex structures [9]. K-D Tree NN is often used as the nearest neighbor algorithm within Chamfer Distance, like we described in fig. 6.5, and is used bidirectionally with the following equation:

$$Dist(A, B) + Dist(B, A)$$

where as K-D Tree NN computes only one direction with this equation:

$$Dist(A, B)$$

We apply the K-D Tree NN distance computations on our point clouds for distance measuring of all the points residing inside two different point clouds.

6.2.2 Chamfer Distance Algorithm

The chamfer distance method is a technique used to measure the similarity between two sets of points in a multi-dimensional space. The method works by finding the distance between each point in one set to the nearest point in the other set, and then summing these distances over all points in both sets. The resulting value is the chamfer distance between the two sets. The chamfer distance method is a method for comparing two sets of points in multiple dimensions. It has various applications in fields such as robotics, computer graphics, and object recognition. [67].

$$Chamfer(X,Y) = \sum_{x \in X} \min_{y \in Y} ||x - y||_2^2 + \sum_{y \in Y} \min_{x \in X} ||x - y||_2^2$$

Figure 6.5: The mathematical formula for chamfer distance where X and Y are point clouds with XYZ coordinates [3].

6.2.3 Hausdorff Distance Algorithm

The Hausdorff distance method is a mathematical technique used to measure the similarity between two sets of points in a metric space. It is named after mathematician Felix Hausdorff and is also sometimes referred to as the Hausdorff metric or Hausdorff distance.

The Hausdorff distance method involves calculating the maximum distance between each point in one set and its nearest point in the other set. This process is then repeated in the opposite direction, with the maximum distance between each point in the second set and its nearest point in the first set being calculated. The larger of these two values is then taken as the Hausdorff distance between the two sets. The Hausdorff distance method has applications in a variety of fields, including image processing, computer vision, and pattern recognition. It is commonly used to compare images or other geometric shapes to determine how similar or dissimilar they are. For example, it can be used to compare a digital image of a person's face to a database of known faces to determine if a match exists.

One of the benefits of the Hausdorff distance method is that it is relatively easy to compute and can be applied to a wide range of data types. However, it is important to note that it can be sensitive to outliers and noise in the data, and it may not always provide the most accurate or appropriate measure of similarity. As with any analytical method, it is important to carefully consider the specific needs and characteristics of the data being analyzed when choosing whether to use the Hausdorff distance method or another approach. In conclusion, the Hausdorff distance method is a powerful tool for measuring the similarity between two sets of points in a metric space. Its broad range of applications and ease of use make it a valuable tool for researchers and practitioners in many fields, but careful consideration of its strengths and limitations is essential to ensure accurate and meaningful results [2].

$$Hausdorff(X,Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x,y), \sup_{y \in Y} \inf_{x \in X} d(x,y) \right\}$$

Figure 6.6: The mathematical formula for hausdorff distance where X and Y are point clouds of XYZ, and sup and inf are the supremum and infimum respectively [4].

6.3 Single Blendshape Distance Computation

The following sections will detail results obtained from running point cloud distance measurement algorithms on extracted points from the obtained facial landmarks of our subjects described in chapter 4. For single blendshape distance computation, mouthSmileLeft seen in fig. 6.8b was our selected blendshape, as this particular blendshape should be able to distinguish between a smiling and non-smiling face. The blendshape mouthSmileLeft is used in combination with the neutral blendshape depicted in fig. 6.8a and a captured face of a subject for all subsequent blendshape weight computations, as depicted in fig. 6.1 . Three subjects from the dataset we created in chapter 4 were chosen for these evaluations: 68002, 68034, 68037 as depicted in figs. 6.7a to 6.7c. The first two subjects are smiling while the last one is of a non-smiling nature.

In order to evaluate the precision of the computed blendshape weight results, we need to visually estimate the expected blendshape weights for each subject involved in single blendshape computation, as there exists no other available approach to assess these outcomes, to the best of our knowledge.

We apply the blendshape weight calculation described in eq. (6.1) with point clouds extracted from a captured face, neutral and target blendshape, to obtain the blendshape weight. The weight is a floating point number between 0 and 1, representing how much a captured face is expressing facial muscle movements that are present in a blendshape, as illustrated in fig. 6.1. For each of the subjects, as seen in figs. 6.7a to 6.7c, we visually predicted the *expected* blendshape weight of blendshape mouthSmileLeft seen in fig. 6.2b.

Each of the subject facial captures were arranged on a vertical line with a neutral and fully expressed blendshape for comparison. This was to get a better visualization of the subject faces relative to the neutral and target blendshape when estimating the blendshape weight, similar to what can be seen in fig. 6.1.

In fig. 6.9 we observed that the mouth of the captured face looks more like blendshape mouthSmileLeft than the neutral blendshape, indicating that a calculated blendshape weight should be in the range $\approx 0.6 - 1.0$. Comparisons seen in fig. 6.10 indicated that the mouth of the captured face appeared to be between in the middle, between the neutral and target blendshape mouthSmileLeft, indicating that a blendshape weight between $\approx 0.4 - 0.8$ would be within our expectations. Lastly, fig. 6.11 showed that the mouth of the captured face looked noticeably more like the neutral face than the blendshape mouthSmileLeft. We expected a blendshape weight in the range between $\approx 0.0 - 0.2$, for that particular subject. The resulting blendshape weights that are derived from the distance computation in this section are evaluated against these expected blendshape weights.

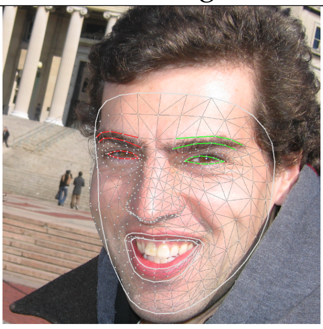
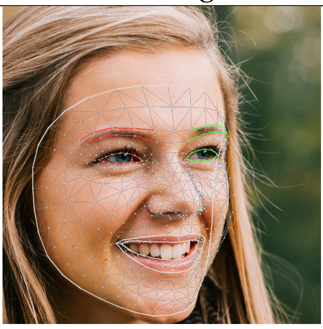
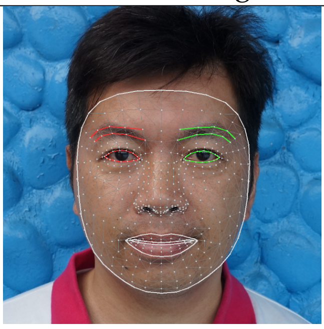
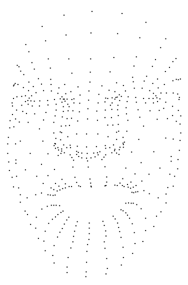
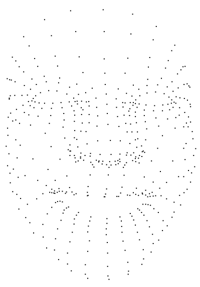
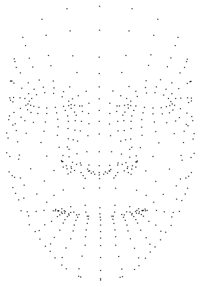
Smiling	Smiling	Non-smiling
 (a) Subject 68002	 (b) Subject 68034	 (c) Subject 68037
		

Figure 6.7: The three subjects used in single blendshape distance computations and their respective point clouds.

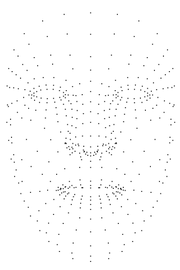
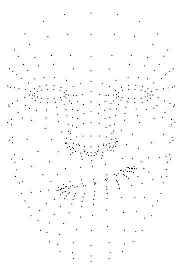
 (a) Neutral	 (b) mouthSmileLeft

Figure 6.8: Point cloud representation of blendshapes neutral seen in fig. 6.8a and mouthSmileLeft seen in fig. 6.8b used in single blendshape distance computations.

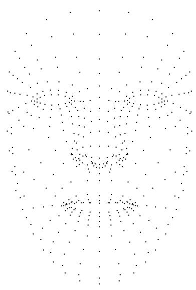
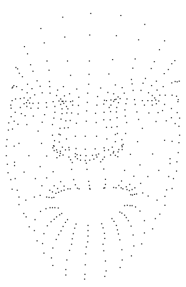
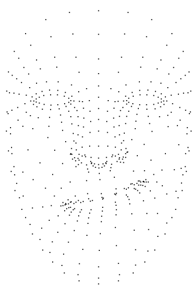
Neutral	Captured	Target Blendshape
		

Figure 6.9: The neutral blendshape, captured face (68002) and fully expressed blendshape mouthSmileLeft as point clouds. Expected blendshape weight: $\approx 0.6 - 1.0$.

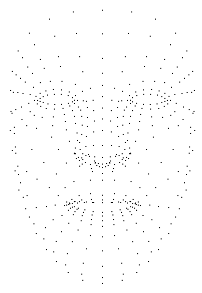
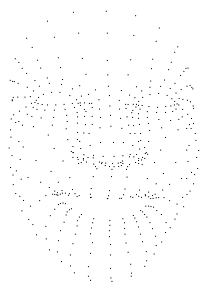
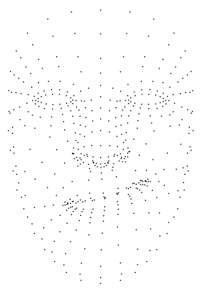
Neutral	Captured	Target Blendshape
		

Figure 6.10: The neutral blendshape, captured face (68034) and fully expressed blendshape mouthSmileLeft as point clouds. Expected blendshape weight: $\approx 0.4 - 0.8$.

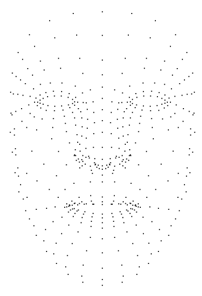
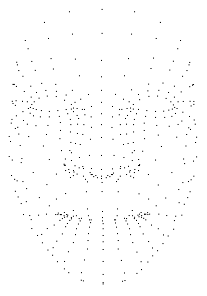
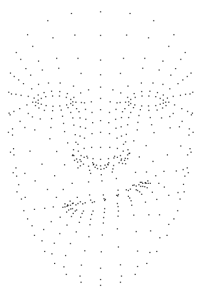
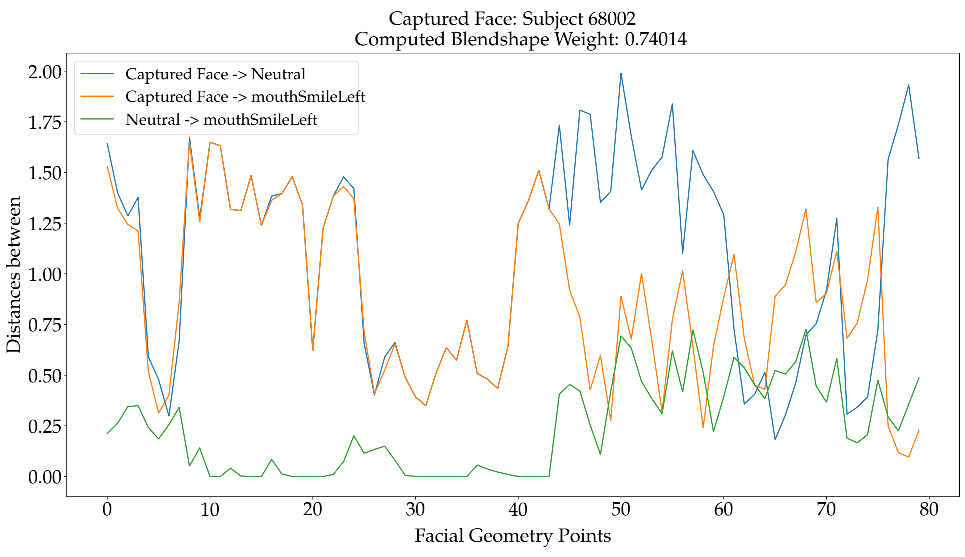
Neutral	Captured	Target Blendshape
		

Figure 6.11: The neutral blendshape, captured face (68037) and fully expressed blendshape mouthSmileLeft as point clouds. Expected blendshape weight: $\approx 0.0 - 0.2$.

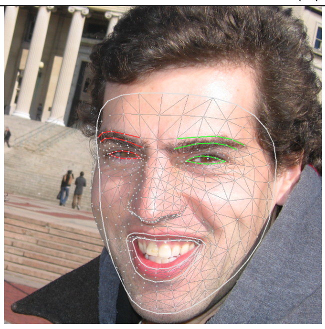
6.4 Single Blendshape: K-D Tree NN distance

The K-D Tree NN distance measurement algorithm was applied to compute the point cloud distances, using the formula in eq. (6.1) between the neutral blendshape from fig. 6.2a, a captured face, and a target blendshape as depicted in fig. 6.2b. The computed distances were turned into blendshape weights which were applied to a 3D avatar to visualize the computed blendshape weight equivalent facial expression.

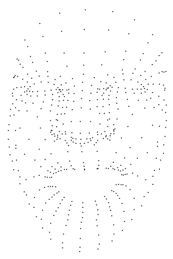
Results in fig. 6.13 show the point cloud distances measured of the captured face, as depicted in fig. 6.7a using K-D Tree NN. As we observe, the distances between the mouths of neutral face and mouthSmileLeft (in green) is relatively small when compared to distances between captured face and mouthSmileLeft. We interpret this as a reflection of the computed blendshape weight of 0.74014 - meaning that this particular subject's face has an active facial expression in the lower left mouth area.



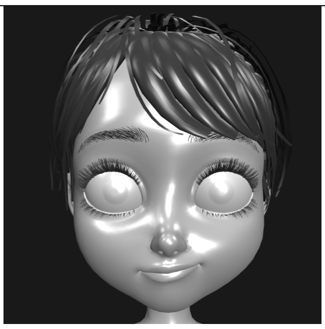
(a) Distances measured for subject 68002.



(b) Captured landmarks of subject 68002.

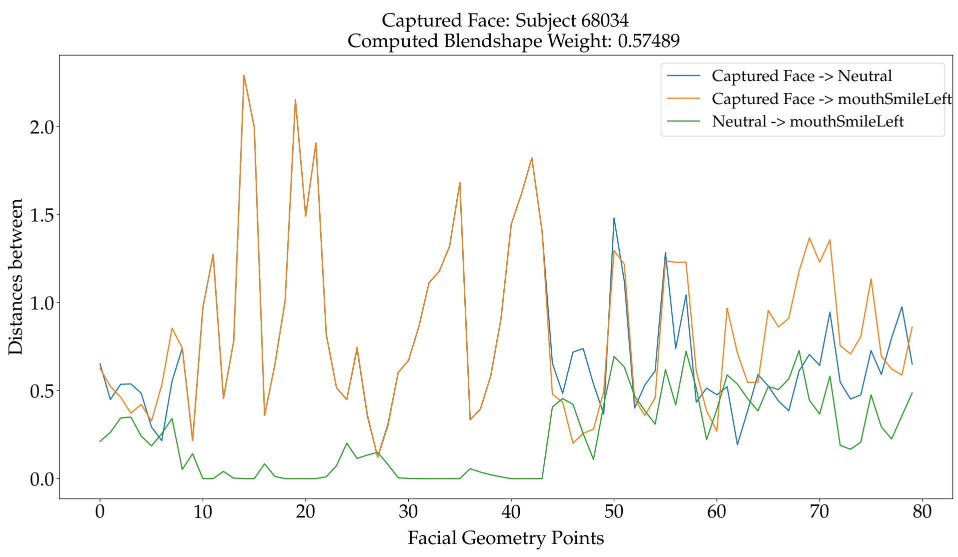


(c) Point cloud of subject 68002.

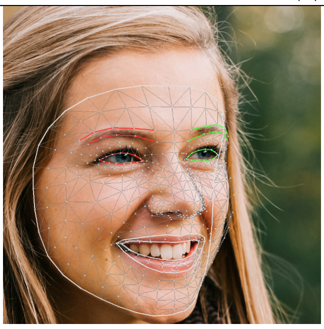


(d) Avatar with blendshape weight applied.

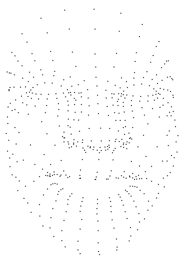
Figure 6.12: Point cloud seen in fig. 6.12c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.74014 is derived from the K-D Tree NN distance computations seen in fig. 6.12a. The blendshape weight is applied to avatar as seen in fig. 6.12d.



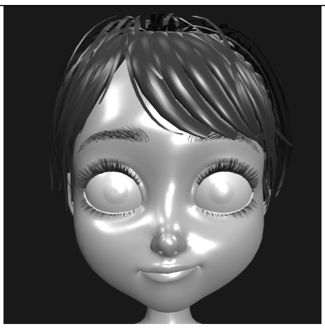
(a) Distances measured for subject 68034.



(b) Captured landmarks of subject 68034.

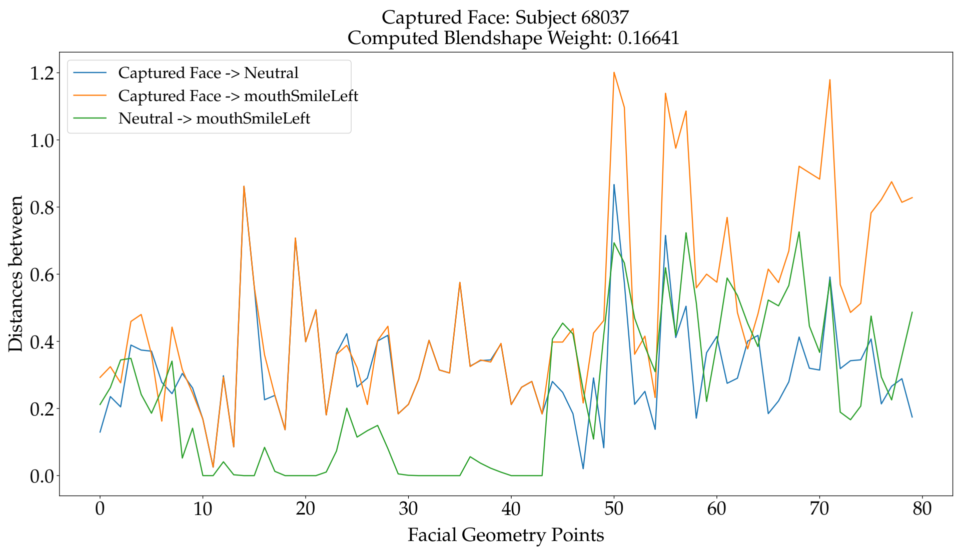


(c) Point cloud of subject 68034.

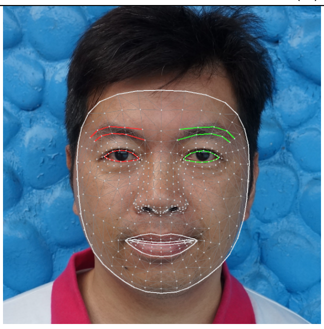


(d) Avatar with blendshape weight applied.

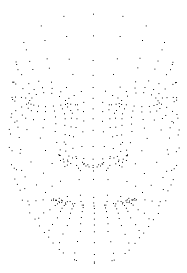
Figure 6.13: Point cloud seen in fig. 6.13c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.57489 is derived from the K-D Tree NN distance computations seen in fig. 6.13a. The blendshape weight is applied to avatar as seen in fig. 6.13d.



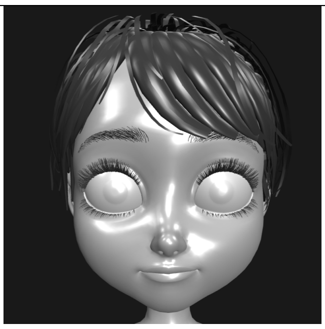
(a) Distances measured for subject 68037.



(b) Captured landmarks of subject 68037.



(c) Point cloud of subject 68037.

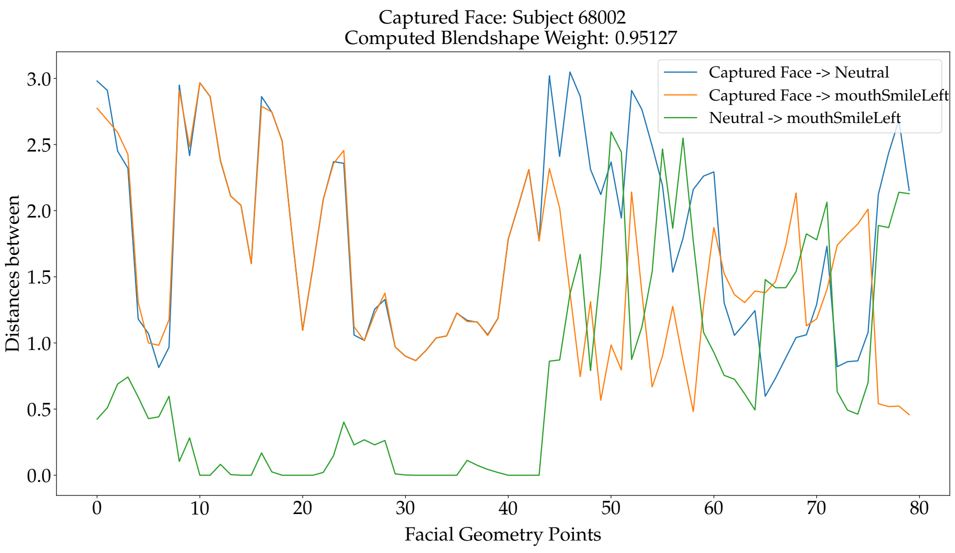


(d) Avatar with blendshape weight applied.

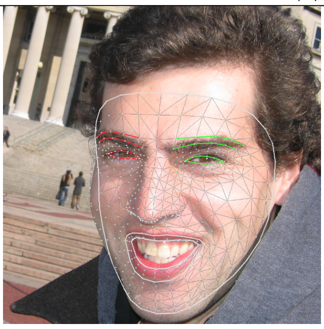
Figure 6.14: Point cloud seen in fig. 6.14c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.16641 is derived from the K-D Tree NN distance computations seen in fig. 6.14a. The blendshape weight is applied to avatar as seen in fig. 6.14d.

6.5 Single Blendshape: Chamfer distance

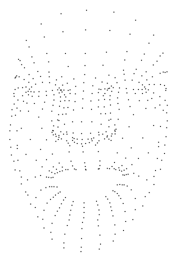
Chamfer Distance measurement results were computed between a captured face, neutral face, and blendshape face, as depicted in eq. (6.1). Results are shown as histogram figures with the computed distances. The computed blendshape weights are applied to avatar in our model viewer.



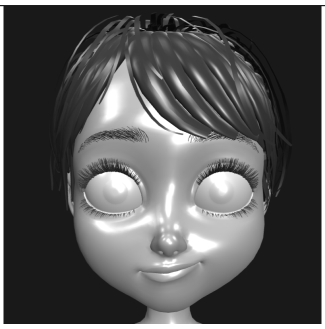
(a) Distances measured for subject 68002.



(b) Captured landmarks of subject 68002.

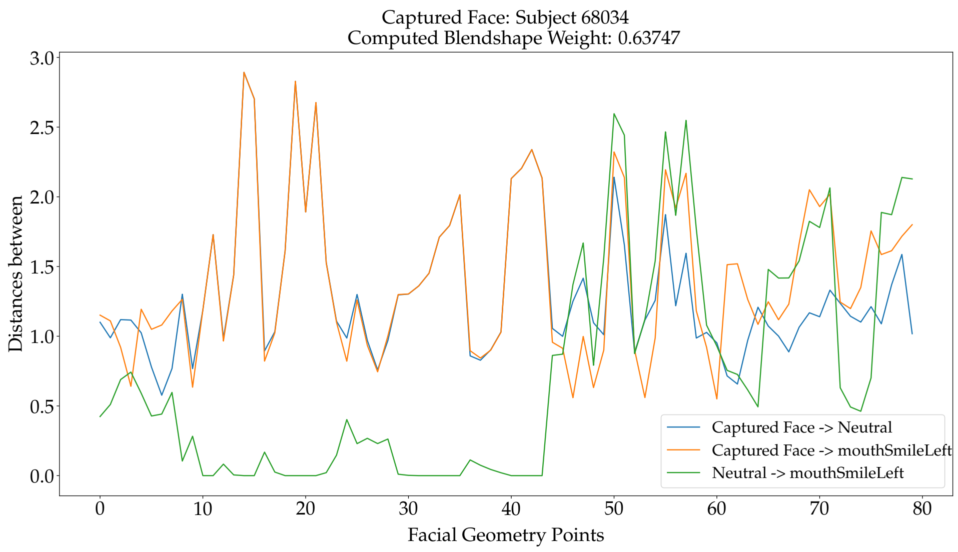


(c) Point cloud of subject 68002.

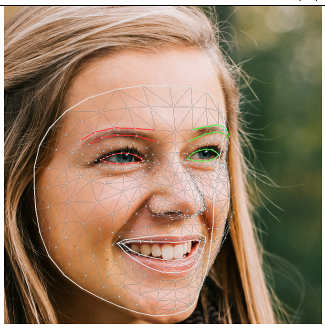


(d) Avatar with blendshape weight applied.

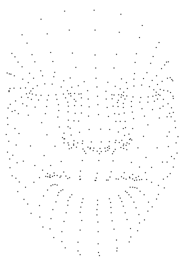
Figure 6.15: Point cloud seen in fig. 6.15c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.95126 is derived from the Chamfer distance computations seen in fig. 6.15a. The blendshape weight is applied to avatar as seen in fig. 6.15d.



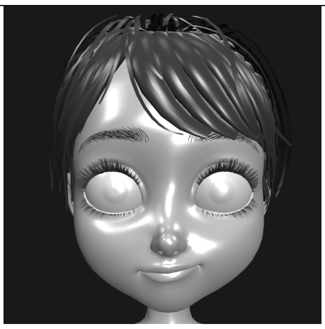
(a) Distances measured for subject 68034.



(b) Captured landmarks of subject 68034.

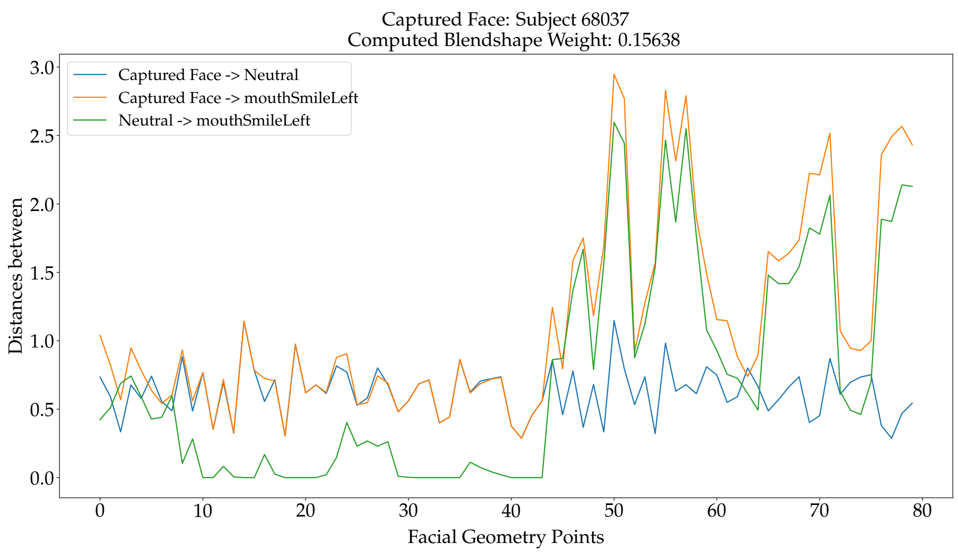


(c) Point cloud of subject 68034.

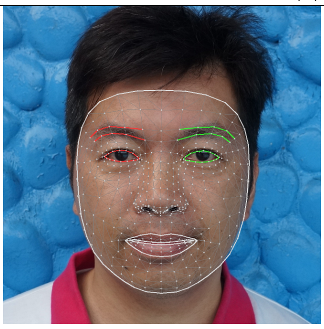


(d) Avatar with blendshape weight applied.

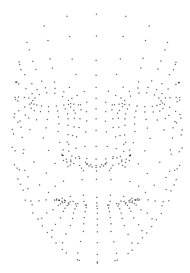
Figure 6.16: Point cloud seen in fig. 6.16c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.63747 is derived from the Chamfer distance computations seen in fig. 6.16a. The blendshape weight is applied to avatar as seen in fig. 6.16d.



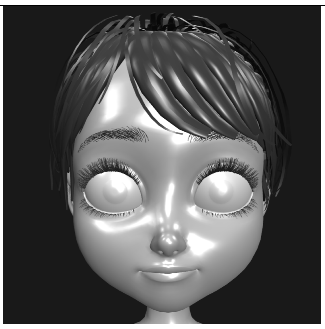
(a) Distances measured for subject 68037.



(b) Captured landmarks of subject 68037.



(c) Point cloud of subject 68037.

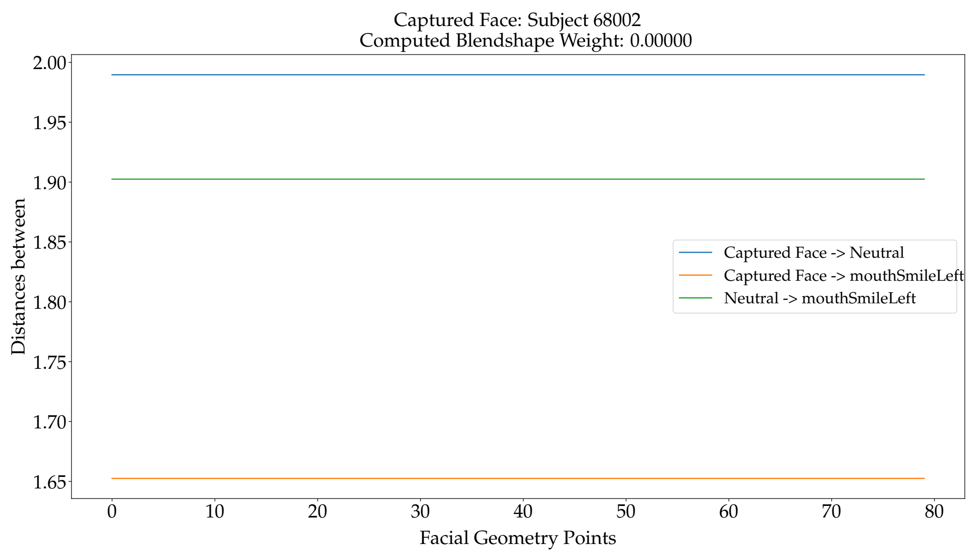


(d) Avatar with blendshape weight applied.

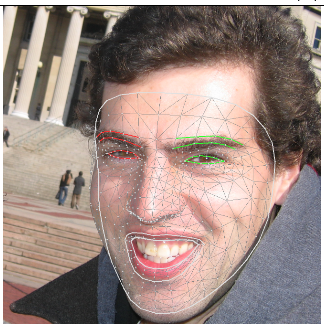
Figure 6.17: Point cloud seen in fig. 6.17c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.15637 is derived from the Chamfer distance computations seen in fig. 6.17a. The blendshape weight is applied to avatar as seen in fig. 6.17d.

6.6 Single Blendshape: Hausdorff distance

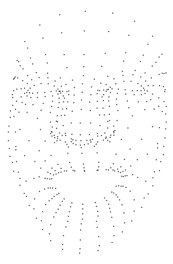
Hausdorff Distance measurement results are computed between captured and neutral face, captured and blendshape face and neutral and blendshape face, as depicted in eq. (6.1). Results are shown as plot figures with the computed distances. The computed blendshape weight is applied to avatar in the model viewer.



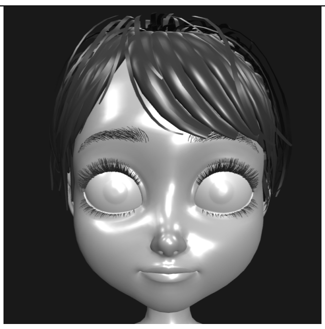
(a) Distances measured for subject 68002.



(b) Captured landmarks of subject 68002.

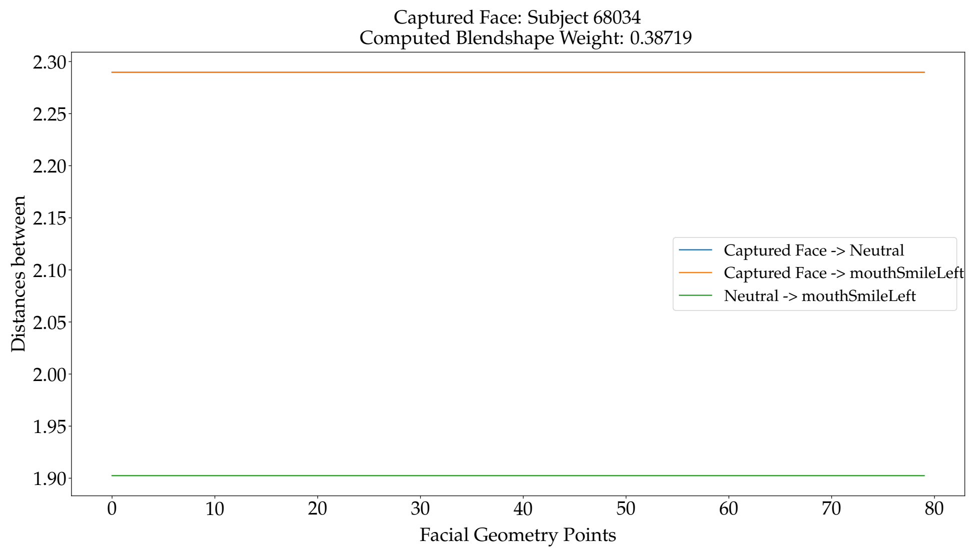


(c) Point cloud of subject 68002.

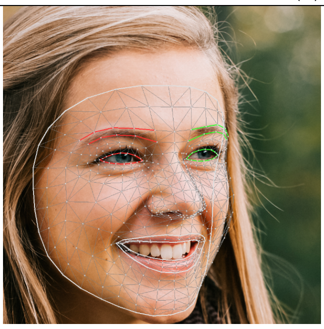


(d) Avatar with blendshape weight applied.

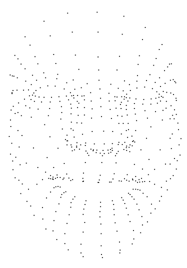
Figure 6.18: Point cloud seen in fig. 6.18c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.15637 is derived from the hausdorff distance computations seen in fig. 6.18a. The blendshape weight is applied to avatar as seen in fig. 6.18d.



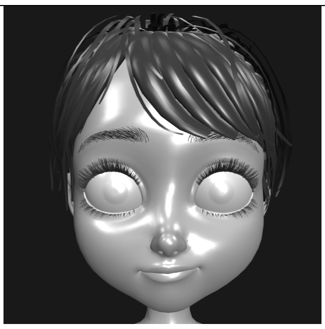
(a) Distances measured for subject 68034.



(b) Captured landmarks of subject 68034.

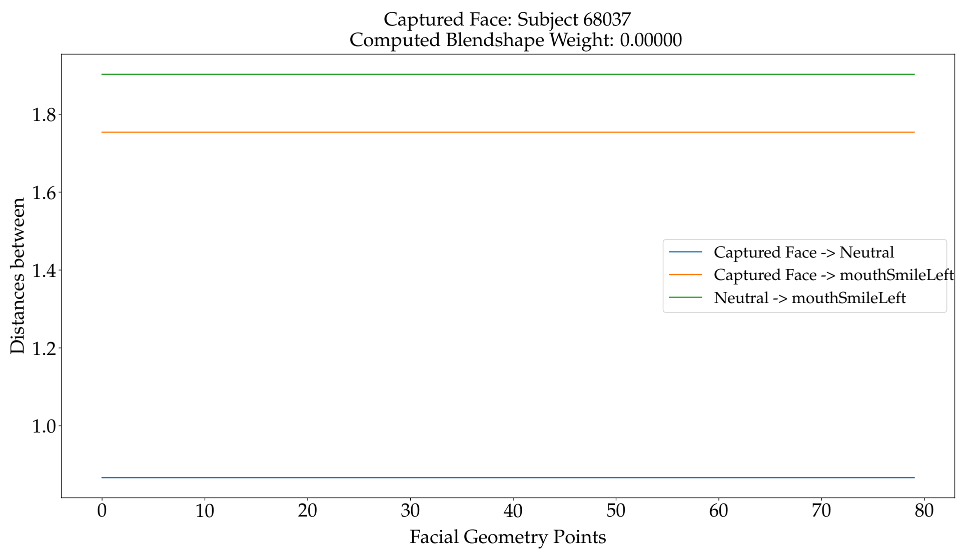


(c) Point cloud of subject 68034.

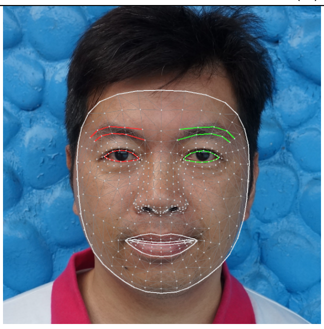


(d) Avatar with blendshape weight applied.

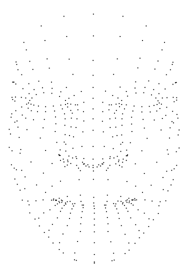
Figure 6.19: Point cloud seen in fig. 6.19c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.38719 is derived from the hausdorff distance computations seen in fig. 6.19a. The blendshape weight is applied to avatar as seen in fig. 6.19d.



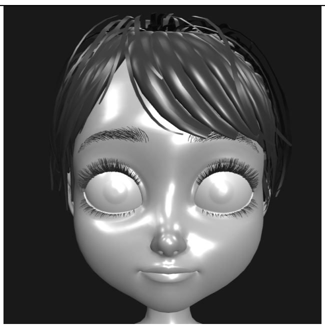
(a) Distances measured for subject 68037.



(b) Captured landmarks of subject 68037.



(c) Point cloud of subject 68037.



(d) Avatar with blendshape weight applied.

Figure 6.20: Point cloud seen in fig. 6.20c are computed against blendshape mouthSmileLeft seen in fig. 6.2b. The calculated blendshape weight of 0.0 is derived from the hausdorff distance computations seen in fig. 6.20a. The blendshape weight is applied to avatar as seen in fig. 6.20d.

6.6.1 Single Blendshape: Summary

Single blendshape weight (mouthSmileLeft) results				
Subject	Expected	K-D-Tree-NN	Chamfer	Hausdorff
68002	$\approx 0.6 - 1.0$	**0.74014**	0.95127	0.00000
68034	$\approx 0.4 - 0.8$	**0.57489**	0.63747	0.38719
68037	$\approx 0.0 - 0.2$	**0.16641**	0.15638	0.00000

Table 6.1: Point cloud distance calculation results using K-D Tree NN, Chamfer and Hausdorff measurement algorithms for single blendshape weight computations: mouthSmileLeft. Bold text indicates the best results.

The experiments for single blendshape demonstrated that by utilizing both K-D Tree NN and Chamfer distance measuring algorithms to compute distances between point clouds, we were able to derive a weight value for the blendshape mouthSmileLeft, that closely resembled the left smile seen in the facial landmark captures from our three test subjects; and fell within our predetermined range of values.

K-D Tree NN clearly showed the best results with all three subjects, as we can see in results table 6.1, however, since Chamfer distance also yielded results within the expected weight range, we deem this an alternative distance measuring technique for weight calculations of blendshape mouthSmileLeft.

The results show that the Hausdorff distance algorithm produced distance measurements with unexpected results, unable to produce results within the expected blendshape weights.

In conclusion, we see promising results using distance measuring algorithms for blendshape weight computations in a single blendshape. However, we recommend implementing additional functionality to support the computation of multiple blendshape weights instead of only a single blendshape weight. As we introduce multiple blendshapes for weight computations, it will increase the evaluation accuracy and the results will show methodology relevance for automatic blendshape weight annotations in datasets.

6.7 Multiple Blendshape Distance Computations

This section shows the details and results of point cloud distance measuring algorithms including K-D Tree NN, Chamfer and Hausdorff, as described in sections 6.2.1 to 6.2.3 respectively. Instead of focusing on a single blendshape (mouthSmileLeft) as done previously in section 6.3, we concentrate on the 24 mouth-related blendshapes seen in fig. 6.22. The distance measurement detailed in eq. (6.1) is applied, and this process is repeated for all distance algorithms. Additionally, we increase our subject count from three to six subjects, depicted in fig. 6.21. Three subjects are smiling, as we see in figs. 6.21a to 6.21c, and the other three subjects are non-smiling, as seen in figs. 6.21d to 6.21f. These subjects are a subset of facial images with landmark captures created in chapter 4.

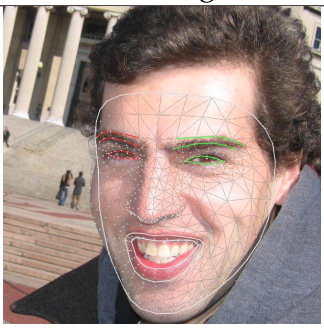
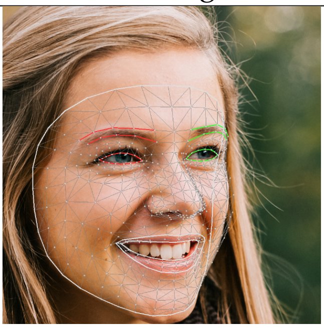
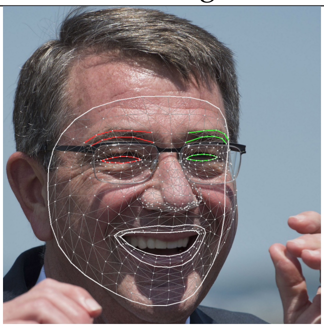
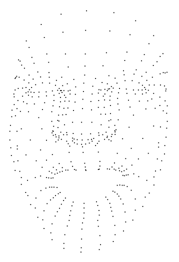
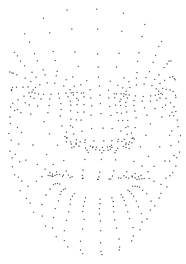
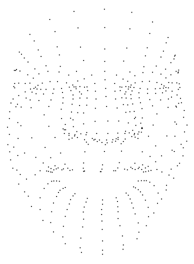
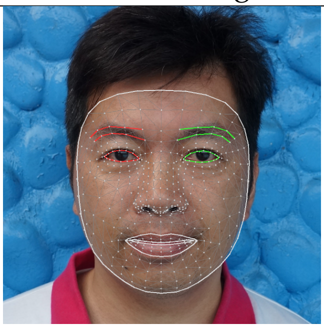
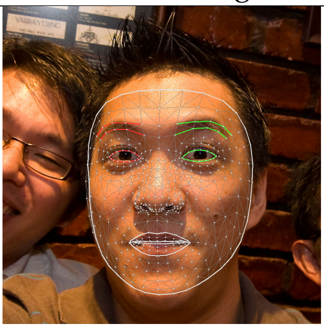
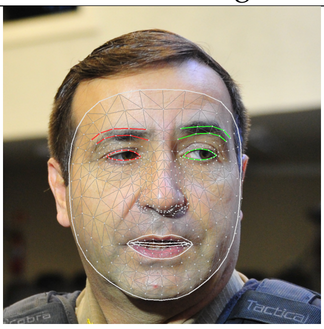
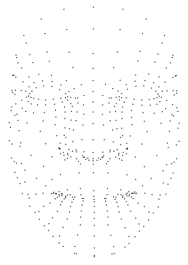
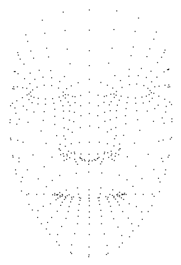
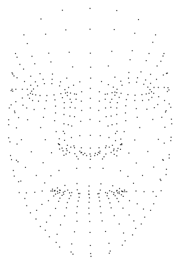
Smiling	Smiling	Smiling
 (a) 68002	 (b) 68034	 (c) 68084
		
Non-smiling	Non-smiling	Non-smiling
 (d) 68037	 (e) 68015	 (f) 68041
		

Figure 6.21: Captured facial landmarks and their respective point clouds of the six subjects used with multiple blendshape distance computations.

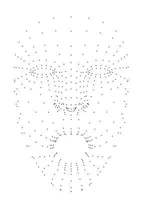
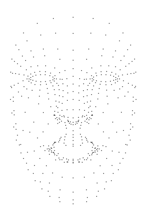
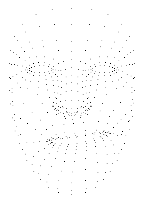
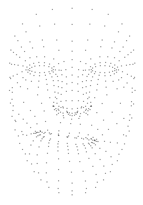
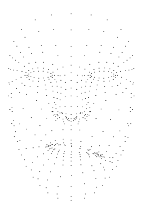
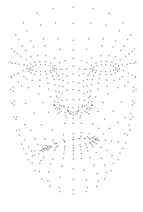
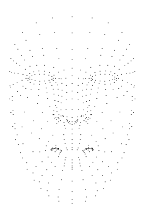
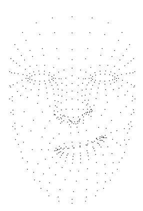
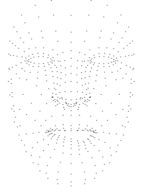
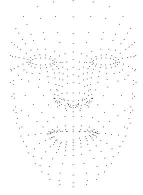
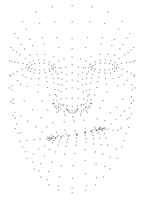
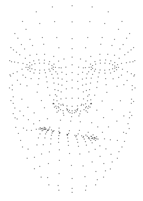
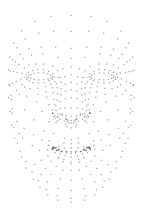
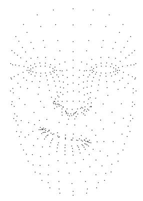
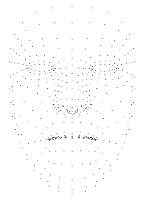
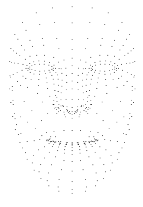
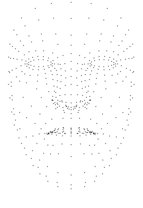
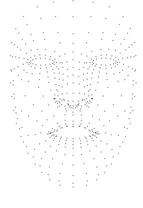
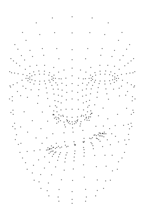
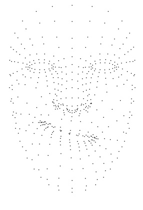
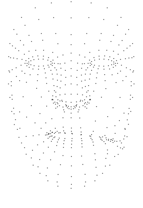
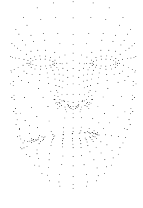
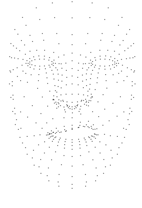
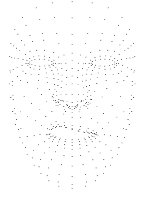
 (a) jawOpen	 (b) mouthClose	 (c) mouthDimpleLeft	 (d) mouthDimpleRight
 (e) mouthFrownLeft	 (f) mouthFrownRight	 (g) mouthFunnel	 (h) mouthLeft
 (i) mouthLowerDownLeft	 (j) mouthLowerDownRight	 (k) mouthPressLeft	 (l) mouthPressRight
 (m) mouthPucker	 (n) mouthRight	 (o) mouthRollLower	 (p) mouthRollUpper
 (q) mouthShrugLower	 (r) mouthShrugUpper	 (s) mouthSmileLeft	 (t) mouthSmileRight
 (u) mouthStretchLeft	 (v) mouthStretchRight	 (w) mouthUpperUpLeft	 (x) mouthUpperUpRight

Figure 6.22: The 24 blendshapes used with multiple blendshape distance computations. Only blendshapes related to mouth movements are used.

6.8 Multiple Blendshape: K-D Tree NN distance

Similar to our previous approach with single blendshape distance computation, we replicate the process by measuring the K-D Tree NN distance, as shown in section 6.2.1, for every mouth-related blendshape in our test; 24 in total, as illustrated in fig. 6.22. We utilize our blendshape weight calculation presented in eq. (6.1) to conduct these measurements.

The results show that all of the subjects, both smiling and non-smiling, as depicted in fig. 6.23, have a high degree of similarity when compared to the mouths of the subjects seen in fig. 6.21.

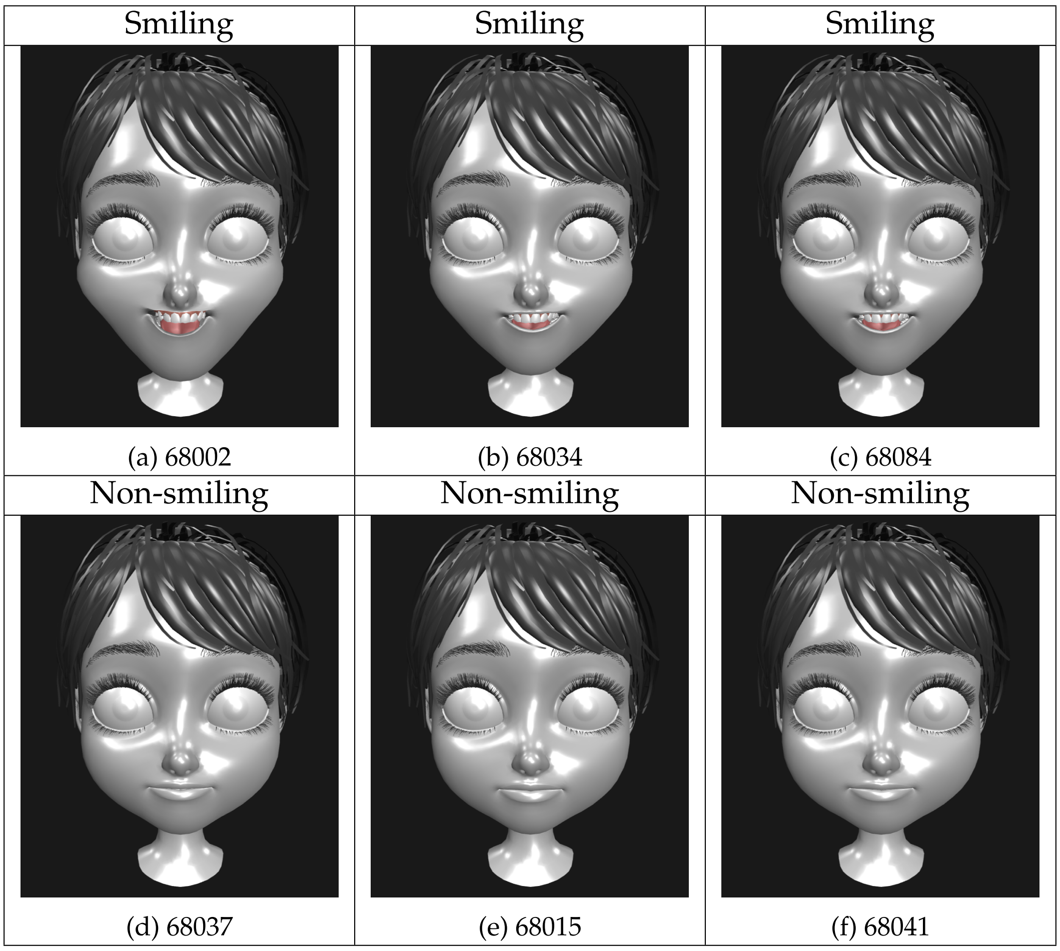


Figure 6.23: Results for K-D Tree NN distance computations for multiple blendshapes. Avatars shown have the calculated blendshape weights for each of the avatar's 24 mouth-related blendshapes applied.

6.9 Multiple Blendshape: Chamfer distance

Utilizing the blendshape weight calculation presented in eq. (6.1), we measure the Chamfer distance, as described in section 6.2.2, for every mouth-related blendshape in our test; a total of 24 as depicted in fig. 6.22.

All three avatars, shown in figs. 6.24a to 6.24c, yield similar poor results for the smiling subjects. Contrarily, the non-smiling subjects shown in figs. 6.24d to 6.24f exhibit considerably better results, with small distance computations, due to the nature of the subjects not smiling.

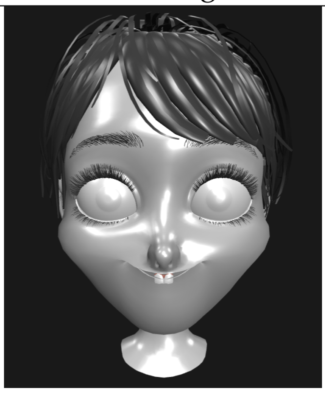
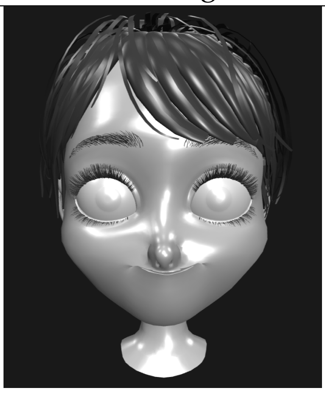
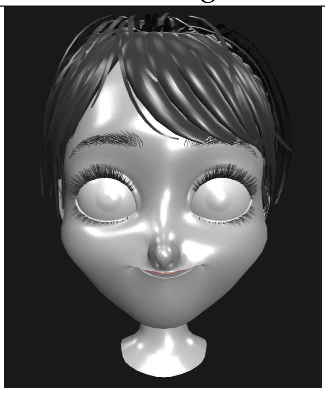
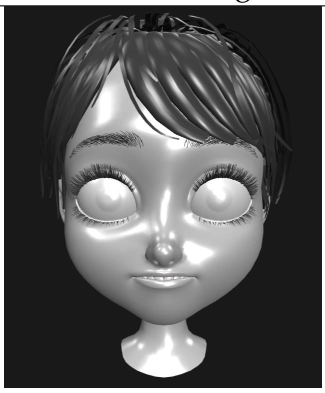
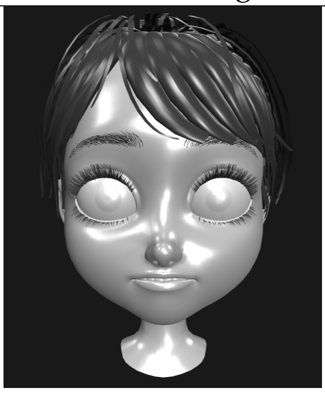
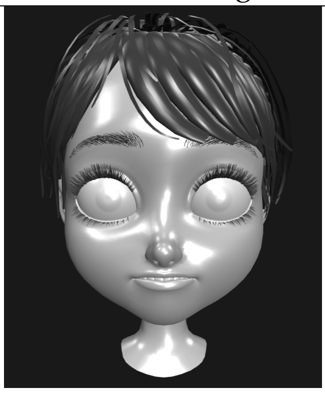
Smiling	Smiling	Smiling
		
(a) 68002	(b) 68034	(c) 68084
Non-smiling	Non-smiling	Non-smiling
		
(d) 68037	(e) 68015	(f) 68041

Figure 6.24: Blendshape weight results from Chamfer distance computations on each of the subject face point clouds. The avatars visualize the resulting blendshape weights calculated from the 24 mouth-related blendshapes, seen in fig. 6.22.

6.10 Multiple Blendshape: Hausdorff distance

Point cloud distance measuring done using the the Hausdorff distance algorithm described in section 6.2.2, combined with our blendshape weight calculation, seen in eq. (6.1), are used to calculate the distances for the selected subjects, depicted in fig. 6.21. The process is repeated for the 24 mouth-related blendshapes seen in fig. 6.22. The resulting blendshape weights are applied to avatars for visual evaluation.

The non-smiling subjects seen in figs. 6.25d to 6.25f, all showed a satisfactory outcome when viewing the avatars. In terms of similarity with the subject faces, the distances measured in non-smiling subjects were very small, but they produced blendshape weights and avatars that looked similar to their subjects. We point out that these results have the same issue found in fig. 6.24, where the small distances seen in non-smiling subjects looks to be accurate, but we see their inaccuracies when applied to faces with expressions.

We can visually confirm that the resulting avatars for the smiling subjects all have similar, almost identical, unsatisfactory outcomes, as displayed in figs. 6.25a to 6.25c. The resulting avatars had little to no resemblance to their respective subjects, despite the avatar results of the non-smiling subjects showing promising results.

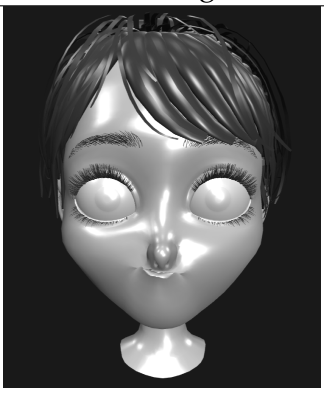
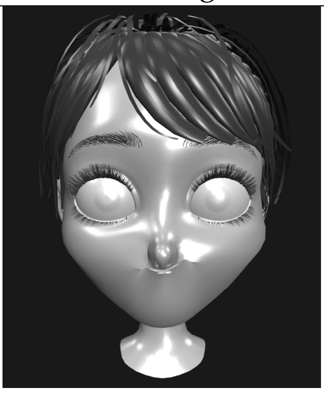
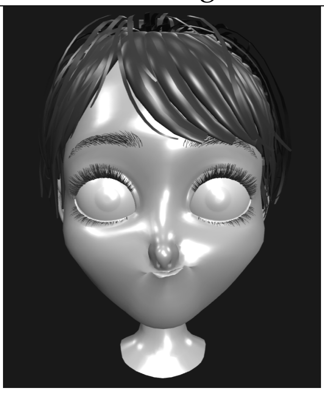
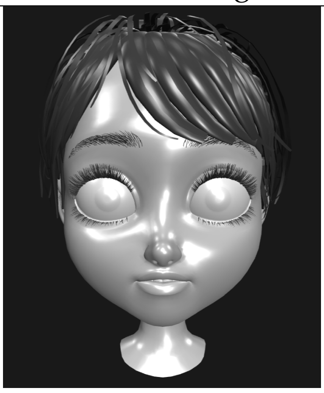
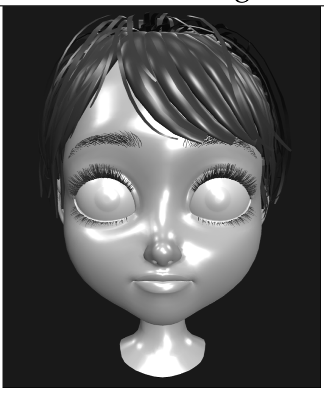
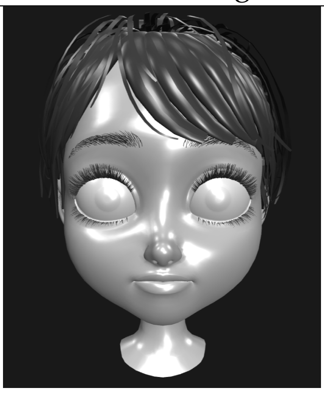
Smiling	Smiling	Smiling
 (a) 68002	 (b) 68034	 (c) 68084
Non-smiling	Non-smiling	Non-smiling
 (d) 68037	 (e) 68015	 (f) 68041

Figure 6.25: The avatars displayed are the blendshape weight results of Hausdorff distance computations which have the calculated blendshape weights applied for each of the 24 mouth-related blendshapes.

6.11 Summary

In this chapter we explored the methodology for quantifying similarity between facial expressions found in facial landmark captures and blendshapes. We looked at different algorithms to compute blendshape weights for a subset of subject faces from our previously created dataset. The evaluation aimed at comparing the effectiveness of three distance algorithms when applied to point clouds, namely K-D Tree NN, Chamfer and Hausdorff , for measuring distances between the points found in the mouth area of a subject face, a neutral blendshape and mouth-related target blendshapes. We initiated the tests with a single blendshape for weight computations using three test subjects and continued with multiple blendshape weight computations for six test subjects.

When applying the Hausdorff distance algorithm using only a single blendshape for weight computation, one could clearly see a flat line of distances between facial points in the histograms when measuring distances between the three faces (captured face, neutral blendshape, target blendshape). This flat line indicated that the applied Hausdorff algorithm failed to produce any meaningful distances. When applying the same Hausdorff blendshape weight calculations to all of the 24 mouth-related blendshapes, we could visually confirm that the resulting avatars looked nothing like their subject faces. Which in turn meant that the resulting blendshape weight calculations were wrong for both single and multiple blendshapes.

For the Chamfer distance algorithm, we had positive results in single blendshape distance computations, using the mouthSmileLeft blendshape. However, the multiple blendshape distance computation results clearly yield poor results, similar to what we saw with the Hausdorff distance algorithm. For our particular use case, we conclude that whether for the computation of individual blendshape weights or multiple blendshape weights, we consider the utilization of both the Hausdorff and Chamfer algorithm to be ineffective.

The findings indicate that the K-D Tree NN algorithm generated the most visually correct avatars for both single blendshape and multiple blendshape weight computations. Specifically, the use of the K-D Tree NN algorithm for multiple blendshape weight computations led to avatars with mouth expressions that closely resembled those of the subject faces.

After analyzing the outcomes and assessing the findings, we can deduce that the utilization of the K-D Tree NN algorithm for measuring point cloud distance and calculating blendshape weight in point clouds has proven to be more effective than the other algorithms examined in this chapter. With reference to research objective 4 described in section 1.3, we have successfully created a tool that facilitates the annotation of facial landmark captures with mouth-related blendshape weights, which can be utilized in training pipelines for machine learning or facial expression analysis research.

Chapter 7

Conclusion and Future Work

7.1 Summary

Revisiting the problem statement, our main objective was to overcome the challenge of capturing and transferring facial expressions from pre-recorded videos, live video feeds, and facial images using affordable cameras without depth sensors to animate a 3D virtual avatar within the police interview training program.

This thesis aimed at creating a 3D virtual avatar for a training program to educate law enforcement personnel, enabling them to improve their skill-set and conduct well-structured interviews with children who are suspected of being subjected to sexual abuse.

We initially conducted experiments to transfer facial landmarks from pre-recorded videos and webcam feeds to a 3D face mesh and virtual avatar with blendshape support. We found that MediaPipe's facial landmark predictions were unstable and noisy, resulting in inaccurate and blurry expression and movement transfer. To address this, we proposed capturing FACS Action Units and deriving blendshape weights that match these units for more realistic facial movements. The proposed method was implemented as a tool to calculate blendshape weights based on the ratios of eye and mouth opening/closing and successfully applied them to a 3D virtual avatar. However, the results only demonstrate this achievement for basic blendshape animations, and further investigation was needed to extend the research to more intricate blendshape animations. From the observations made during these initial experiments, we suggested generating a dataset for machine learning purposes to learn facial landmark features and predict blendshape weights.

The following work focused on creating a dataset of facial landmarks and 3D blendshape face models for machine learning pipelines to predict blendshape weights. We derived facial landmarks from the FFHQ dataset of facial images and created 3D blendshape face models. To minimize errors encountered, we aligned the face point cloud data with the base blendshape face model's translation, rotation, and scale. We faced difficulties in automating the transfer of facial expressions from the ARKit source models to our models, so we manually sculpted the blendshape face models.

Despite the challenges we faced, our dataset, PointFaces, can be used for further research on predicting blendshapes from pre-recorded videos, live camera feeds or static images.

Following the creation of the dataset, we conducted experiments on machine learning for point clouds. We discussesed the implementation of machine learning architectures using the PointFaces dataset we created to predict blendshape weights. The Siamese One Shot Learning network was used, but it was found to be primarily suited for images and difficult to interpret results with point clouds. PointNet, a state-of-the-art deep learning architecture, was also experimented with, but it required time consuming data pre-processing, and the results were not satisfactory due to missing triangles in the reconstructed facial mesh. We concluded that a more robust approach was needed and suggested exploring specific operations and algorithms for point cloud comparisons and conducting experiments with point cloud specific machine learning architectures.

The work was furthered by implementing and conducting experiments with point clouds to estimate blendshape weights using distance algorithms. We discussed the exploration of different algorithms to compute blendshape weights for facial landmark captures and blendshapes. We compared the effectiveness of the K-D Tree NN, Chamfer, and Hausdorff algorithms for measuring distances between points in the mouth area of a subject face, neutral blendshape, and target blendshapes. The Hausdorff and Chamfer algorithms were found to be ineffective for both single and multiple blendshape weight computations. The K-D Tree NN algorithm produced the most visually correct avatars for both single and multiple blendshape weight computations, closely resembling the subject faces' mouth expressions. We concluded that we had successfully developed a tool for annotating facial landmark captures with mouth-related blendshape weights, which can be utilized in machine learning training pipelines or research on facial expression analysis.

Deduced by the work conducted during this thesis, we observed that our methods holds the potential to animate a 3D virtual avatar in the training program in two distinct ways. Either by a hired actor through the use of a live video feed, or by transfer of facial mimicry captured from an abused child in pre-recorded interview videos. Both of these are viable 3D virtual avatar animation methods, but suffered from inaccuracte expression transfers. The dataset we created, PointFaces, includes face point clouds and 3D blendshape face models. To annotate the dataset with blendshape weights, we developed a tool that automatically annotates blendshape weights matching the facial expressions and movements of the mouth area. The dataset and it's subsequent annotations can be utilized in further research in blendshape weight estimation using machine learning, to predict blendshape weights for use in 3D virtual avatar animation.

7.1.1 Contributions

During the course of this thesis, we conducted research and developed tools for transferring facial expressions from pre-recorded videos, live video feeds, and facial images. Our focus was on the development of a 3D virtual avatar that could be controlled by a hired actor through inexpensive camera equipment, or through facial expressions captured from an abused child in pre-recorded interview videos.

This study is aimed at creating a training program to educate law enforcement personnel, enabling them to improve their skill-set and conduct well-structured interviews with children who are suspected of being subjected to sexual abuse. Our tools progressively increases complexity as we discover findings, drawbacks and limitations, within our experiments.

In this section we go through each of the research objectives we described in the problem statement stated in section 1.3 with a description of how we solved each of the stated objectives. Each objective is supported by experiments conducted using different algorithms and techniques mentioned throughout this thesis.

Objective 1: *Research and develop a proof-of-concept implementation for facial expression transfer from video using facial landmarks.*

The objective is supported by the development of a tool that detects facial landmarks in pre-recorded videos and subsequent transfer of facial landmarks onto a 3D face mesh. Using this tool we can transfer facial expressions from children in pre-recorded child abuse interviews onto a 3D face mesh.

Objective 2: *Research and develop a proof-of-concept implementation for animating 3D virtual avatars using blendshape weights.*

The objective is supported by the development of a tool that calculates the Eye-Aspect-Ratio (EAR) and Mouth-Aspect-Ratio (MAR) for measuring the opening and closing ratios of eyes and mouth, respectively. The ratios are utilized to compute blendshape weights, which are subsequently transferred to a 3D virtual avatar with blendshape support. For visualization support, we create a tool that integrates web camera input feed and displays blendshape names values corresponding to the calcuated eyes and mouth opening and closing.

Objective 3: *Develop a dataset of facial landmarks and 3D blendshape face models for use in machine learning pipelines, and experiment with machine learning architectures utilizing such dataset.*

The development of PointFaces has supported the objective, which is a dataset consisting of facial landmark captures in various forms such as point cloud formats (PLY/PCD) and images (PNG). The dataset is accompanied by a set of sculpted 3D blendshape face models and can be utilized for further research in detecting or predicting various blendshapes from image or video inputs, as well as for facial expression analysis. PointFaces is made publicly available at Github, https://github.com/olealgoritme/pointfaces, and Zenodo, https://zenodo.org/record/7900081.

To further support the objective, we conducted experiments by implementing machine learning architectures such as Siamese One Shot and PointNet, which utilize the produced dataset.

Objective 4: *Research and develop a tool for automatic annotation of blendshape* *weights from facial landmark captures in facial images.*

To support our objective, we have created a proof-of-concept tool that leverages point cloud distance calculations to derive blendshape weights. We achieve this by utilizing K-D Tree Nearest Neighbor calculations on facial landmark captures obtained from our PointFaces dataset, which we developed specifically for predicting blendshape weights and analyzing facial expressions. By utilizing this tool, we can generate a 3D virtual avatar that displays the appropriate blendshape weights based on the input facial image. This work is made publicly available at Github, https://github.com/olealgoritme/master_thesis.

7.1.2 Future Work

Gaining a deeper understanding of the utilization of point cloud data derived from facial landmark captures in the realm of machine learning and artificial intelligence is crucial in developing an accurate blendshape weight prediction model for avatar animations. In chapter 6, we investigated how to quantify the similarity between a neutral and target blendshape using a subject's facial landmarks for blendshape weight approximation. We recognize that the approach had limitations as the manual process of scaling and transforming a subject's captured face to a neutral blendshape face was slow and time consuming. As future work, we aim to automate this process so that any input facial landmarks in any rotation, scale, and format can be transformed to match that of a source blendshape without requiring manual intervention. Optimizing the code using a low-level programming language such as C/C++ would lead to additional enhancements and increased calculation speed of blendshape weights, making it suitable for real-time processing in virtual avatar applications.

We would also like to employ our annotated blendshape weight dataset in point cloud feature learning networks, such as RBFNet, a Radial Basis Function deep learning technique, which has promising features. Unlike PointNet variants, which face challenges in recognizing local point patterns, the RBFNet approach explicitly models the spatial distribution of point clouds by extracting features from sparsely positioned RBF kernels. A typical RBF kernel, such as a Gaussian function, discourages long-distance responses and only responds to neighboring points. This localized response generates highly distinctive features that vary based on the point distributions and are highly discriminative, as described in [14].

Bibliography

[1] Malak Abdullah, Alia Madain, and Yaser Jararweh. "ChatGPT: Fundamentals, Applications and Social Impacts." In: *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. 2022, pp. 1–8. DOI: 10.1109/SNAMS58071.2022.10062688.

[2] Mehboob Ali, Zahid Hussain, and Miin-Shen Yang. "Hausdorff Distance and Similarity Measures for Single-Valued Neutrosophic Sets with Application in Multi-Criteria Decision Making." In: *Electronics* 12.1 (2022), p. 201.

[3] Brad Chambers et al. Andrew Bell. *Chamfer*. Accessed 20-02-2023. 2023. URL: https://pdal.io/en/latest/apps/chamfer.html.

[4] Brad Chambers et al. Andrew Bell. *Hausdorff*. Accessed: 20-02-2023.

[5] *ARKit BlendShape Model Website*. Accessed 25-08-2022. URL: https://arkit-face-blendshapes.com/.

[6] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets." In: *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), pp. 698–700.

[7] Gunn Astrid Baugerud et al. "Multimodal virtual avatars for investigative interviews with children." In: *Proceedings of the 2021 Workshop on Intelligent Cross-Data Analysis and Retrieval*. 2021, pp. 2–8.

[8] Kai Xin Beh and Kam Meng Goh. "Micro-expression spotting using facial landmarks." In: *2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA)*. IEEE. 2019, pp. 192–197.

[9] Jon Louis Bentley. "K-d trees for semidynamic point sets." In: *Proceedings of the sixth annual symposium on Computational geometry*. 1990, pp. 187–197.

[10] Fausto Bernardini et al. "The ball-pivoting algorithm for surface reconstruction." In: *IEEE transactions on visualization and computer graphics* 5.4 (1999), pp. 349–359.

[11] Géry Casiez, Nicolas Roussel, and Daniel Vogel. "1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 2527–2530.

[12] Ann-Christin Cederborg et al. "Investigative interviews of child witnesses in Sweden." In: *Child abuse & neglect* 24.10 (2000), pp. 1355–1361.

[13] Stanford University Charles R. Qi et al. *PointNet*. Accessed 20-12-2022. 2017. URL: https://stanford.edu/~rqi/pointnet.

[14] Weikai Chen et al. "Deep rbfnet: Point cloud feature learning using radial basis functions." In: *arXiv preprint arXiv:1812.04302* (2018).

[15] Pytorch contributors. *Adam: Pytorch*. Accessed 20-12-2022. 2023. URL: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html.

[16] Threejs contributors. *Threejs, 3D javascript library*. Accessed 03-02-2022. 2023. URL: https://github.com/mrdoob/three.js.

[17] Wikipedia contributors. *Face (geometry)*. Accessed 21-12-2022. 2022. URL: https://en.wikipedia.org/wiki/Face_(geometry).

[18] Wikipedia contributors. *Iterative Closest Point algorithm*. Accessed 10-12-2022. 2022. URL: https://en.wikipedia.org/wiki/Iterative_closest_point.

[19] Wikipedia contributors. *Normal (geometry)*. Accessed 21-12-2022. URL: https://en.wikipedia.org/wiki/Normal_(geometry).

[20] Wikipedia contributors. *PLY (file-format)*. Accessed 10-12-2022. 2022. URL: https://en.wikipedia.org/wiki/PLY_(file_format).

[21] Wikipedia contributors. *Point Cloud*. Accessed 20-02-2023. 2023. URL: https://en.wikipedia.org/wiki/Point_cloud.

[22] Wikipedia contributors. *The Metaverse*. Accessed 18-04-2023. 2023. URL: https://en.wikipedia.org/wiki/Metaverse.

[23] Ivan Culjak et al. "A brief introduction to OpenCV." In: *2012 proceedings of the 35th international convention MIPRO*. IEEE. 2012, pp. 1725–1730.

[24] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning." In: *Machine learning techniques for multimedia: case studies on organization and retrieval* (2008), pp. 21–49.

[25] Peter J. Denning et al. "Computing as a discipline." In: *Computer* 22.2 (1989), pp. 63–70.

[26] Robert Dorfman. "A formula for the Gini coefficient." In: *The review of economics and statistics* (1979), pp. 146–149.

[27] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. "On the shape of a set of points in the plane." In: *IEEE Transactions on information theory* 29.4 (1983), pp. 551–559.

[28] Paul Ekman and Wallace V Friesen. "Facial action coding system." In: *Environmental Psychology & Nonverbal Behavior* (1978).

[29] Zoubin Ghahramani. "Unsupervised learning." In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures* (2004), pp. 72–112.

[30] Google. *Face Mesh: MediaPipe*. Accessed 15-04-2022. URL: https://google.github.io/mediapipe/solutions/face_mesh.html.

[31] Google. *MediaPipe Canonical Face Model*. Accessed 13-12-2022. URL: https://google.github.io/mediapipe/solutions/face_mesh.html#canonical-face-model.

[32] MediaPipe Google. *FFHQ Dataset by NVIDIA*. Accessed 10-10-2022. 2019. URL: https://github.com/NVlabs/ffhq-dataset.

[33] Arthur H Green. “Child maltreatment and its victims: A comparison of physical and sexual abuse.” In: *Psychiatric Clinics of North America* 11.4 (1988), pp. 591–610.

[34] Arthur H Green. “Dimension of psychological trauma in abused children.” In: *Journal of the American Academy of Child Psychiatry* 22.3 (1983), pp. 231–237.

[35] Ivan Grishchenko et al. “Attention Mesh: High-fidelity Face Mesh Prediction in Real-time.” In: *arXiv preprint arXiv:2006.10962* (2020).

[36] Harshvardhan Gupta. *Facial Similarity with Siamese Networks in PyTorch*. Accessed 18-12-2022. 2017. URL: https://hackernoon.com/facial-similarity-with-siamese-networks-in-pytorch-9642aa9db2f7.

[37] Carl-Herman Hjortsjö. *Man's face and mimic language*. Studentlitteratur, 1969.

[38] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. “PyTorch.” In: *Programming with TensorFlow: Solution for Edge Computing Applications* (2021), pp. 87–104.

[39] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[40] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson surface reconstruction.” In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006.

[41] Brian R Kent. *3D scientific visualization with Blender®*. Morgan & Claypool Publishers, 2015.

[42] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. “Siamese neural networks for one-shot image recognition.” In: *ICML deep learning workshop*. Vol. 2. Lille. 2015.

[43] Martin Koestinger et al. “Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization.” In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 2144–2151.

[44] Julia Korkman, Pekka Santtila, and N Kenneth Sandnabba. “Dynamics of verbal interaction between interviewer and child in interviews with alleged victims of child sexual abuse.” In: *Scandinavian journal of psychology* 47.2 (2006), pp. 109–119.

[45] Akihiro Kuwahara et al. “Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping (EARM).” In: *Cognitive Robotics* 2 (2022), pp. 50–59.

[46] Myrthe Lammerse. *Text-based emotion detection to create a virtual avatar interview-training program*. Accessed 9-06-2022. 2022. URL: http://home.simula.no/~paalh/students/MyrtheLammerse-UiO-2022.pdf.

[47] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. “Realistic modeling for facial animation.” In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 1995, pp. 55–62.

[48] Ying Li et al. “Deep learning for lidar point clouds in autonomous driving: A review.” In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2020), pp. 3412–3432.

[49] Point Cloud Library. *The PCD File Format*. Accessed 18-04-2023. 2023. URL: https://pointclouds.org/documentation/tutorials/pcd_file_format.html.

[50] Don McCurdy. *glTF-viewer*. Accessed 03-02-2022. 2023. URL: https://github.com/donmccurdy/three-gltf-viewer.

[51] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. “Head pose estimation in computer vision: A survey.” In: *IEEE transactions on pattern analysis and machine intelligence* 31.4 (2008), pp. 607–626.

[52] World Health Organization. *Pexel Doctor interviewing patient*. Accessed 26-05-2022. 2020. URL: https://www.pexels.com/video/a-doctor-interviewing-the-patient-5998346/.

[53] Zainab Oufqir, Abdellatif El Abderrahmani, and Khalid Satori. “ARKit and ARCore in serve to augmented reality.” In: *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. IEEE. 2020, pp. 1–7.

[54] Divyarajsinh N Parmar and Brijesh B Mehta. “Face recognition methods & applications.” In: *arXiv preprint arXiv:1403.0485* (2014).

[55] Houwen Peng et al. “RGBD salient object detection: A benchmark and algorithms.” In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III 13*. Springer. 2014, pp. 92–109.

[56] Polywink. *Polywink Sample Character*. Accessed 18-03-2022. 2022. URL: https://www.polywink.com/15-facial-animation-for-iphone-x.html.

[57] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[58] Sanjeev Jagannatha Rao, Yufei Wang, and Garrison W Cottrell. “A Deep Siamese Neural Network Learns the Human-Perceived Similarity Structure of Facial Expressions Without Explicit Categories.” In: *CogSci*. 2016.

[59] Guido van Rossum. *Python3 Whats New*. Accessed: 22-03-2023. URL: https://docs.python.org/3/whatsnew/3.0.html.

[60] Jason M Saragih, Simon Lucey, and Jeffrey F Cohn. “Real-time avatar animation from a single image.” In: *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*. IEEE. 2011, pp. 117–124.

[61] Burcu Kır Savaş and Yaşar Becerikli. “Real time driver fatigue detection based on SVM algorithm.” In: *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*. IEEE. 2018, pp. 1–4.

[62] CP Sumathi, T Santhanam, and M Mahadevi. “Automatic facial expression analysis a survey.” In: *International Journal of Computer Science and Engineering Survey* 3.6 (2012), p. 47.

[63] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." In: *Computing in science & engineering* 13.2 (2011), pp. 22–30.

[64] Daniel Woldegiorgis. "Mimicking Facial Expression from Actor to Virtual Avatar using Machine Learning." In: (2021).

[65] James M Wood and Sena Garven. "How sexual abuse interviews go astray: Implications for prosecutors, police, and child protection services." In: *Child Maltreatment* 5.2 (2000), pp. 109–118.

[66] Baoxin Wu, Chunfeng Yuan, and Weiming Hu. "Human action recognition based on context-dependent graph kernels." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2609–2616.

[67] Tong Wu et al. "Density-aware chamfer distance as a comprehensive metric for point cloud completion." In: *arXiv preprint arXiv:2111.12702* (2021).

[68] Delina Beh Mei Yin et al. "Fusion of face recognition and facial expression detection for authentication: a proposed model." In: *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. 2017, pp. 1–8.

[69] Fan Zhang et al. "Mediapipe hands: On-device real-time hand tracking." In: *arXiv preprint arXiv:2006.10214* (2020).

[70] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A modern library for 3D data processing." In: *arXiv preprint arXiv:1801.09847* (2018).